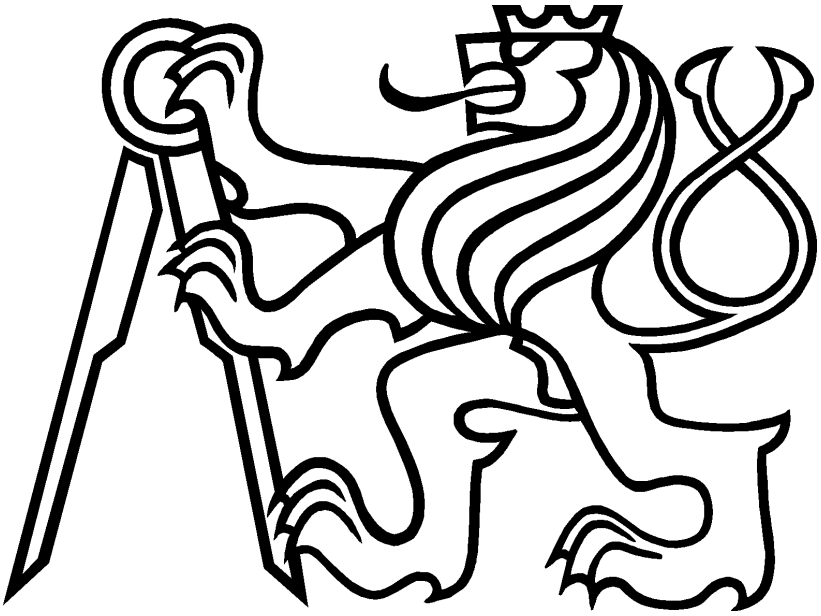
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

# DIPLOMA THESIS



Bc. Vladimír Petrík

## Multi-Vehicle Random Finite Set SLAM

**Department of Cybernetics**

Thesis supervisor: **RNDr. Miroslav Kulich, Ph.D.**

**Czech Technical University in Prague**
**Faculty of Electrical Engineering**

**Department of Cybernetics**

# DIPLOMA THESIS ASSIGNMENT

**Student:** Bc. Vladimír P e t r í k

**Study programme:** Cybernetics and Robotics

**Specialisation**: Robotics

**Title of Diploma Thesis:** Multi-Robot Simultaneous Localization and Mapping

**Guidelines:**

1. Get acquainted with random finite sets theory and with methods of multi-robot simultaneous localization and mapping (SLAM).
2. Develop a multi-robot SLAM method based on random finite sets for a sensor generating 3D point clouds. Implement particular parts of the algorithm as ROS (Robot Operating System, http://ros.org) nodes.
3. Document the resulting code carefully.
4. Verify functionality of the implemented algorithms experimentally. Describe the obtained results with respect to the algorithms' speed and robustness as well as quality of the generated output.

**Bibliography/Sources:**

[1] J. Mullane, B.-N. Vo, M. Adams, B.-T. Vo.: Random Finite Sets for Robot Mapping & SLAM, Springer Tracts in Advanced Robotics 72, Springer-Verlag Berlin Heidelberg, 2011.
[2] A. Doucet, N. de Freitas & N. Gordon (eds): Sequential Monte Carlo Methods in Practice. Springer-Verlag. 2001. ISBN 0-387-95146-6.
[3] S. Thrun, W. Burgard, and D. Fox. Probabilistic Robotics. MIT Press, Cambridge, MA, 2005.
[4] D. Moratuwage B.-N. Vo, and D. Wang, "Collaborative Multi-Vehicle SLAM with Moving Object Tracking", Proc. IEEE Int. Conf. Robotics & Automation, (ICRA'13), Karlsruhe, Germany, May, 2013.
[5] D. Moratuwage, B.-N. Vo, and D. Wang "A Hierarchical Approach to the Multi-Vehicle SLAM Problem", Proc. 15th Annual Conf. Information Fusion, Singapore, 2012.

**Diploma Thesis Supervisor:** RNDr. Miroslav Kulich, Ph.D.

**Valid until:** the end of the summer semester of academic year 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
**Head of Department**

prof. Ing. Pavel Ripka, CSc.
**Dean**

Prague, January 10, 2014

**České vysoké učení technické v Praze**
**Fakulta elektrotechnická**

**Katedra kybernetiky**

# ZADÁNÍ DIPLOMOVÉ PRÁCE

**Student:** Bc. Vladimír P e t r í k

**Studijní program:** Kybernetika a robotika (magisterský)

**Obor:** Robotika

**Název tématu:** Simultánní lokalizace a mapování pro více robotů

**Pokyny pro vypracování:**

1. Seznamte se s teorií konečných náhodných množin a s metodami simultánní lokalizace a mapování (SLAM) pro více robotů.
2. Implementujte metodu SLAM pro více robotů s použitím konečných náhodných množin pro senzor generující 3D mračna bodů. Jednotlivé části algoritmu implementujte jako uzly systému ROS (Robot Operating System, http://ros.org).
3. Kód implementovaných algoritmů důkladně zdokumentujte.
4. Funkčnost implementovaných algoritmů ověřte experimenty. Experimentální výsledky popište se zaměřením na rychlost a robustnost algoritmů a kvalitu generovaných výsledků.

**Seznam odborné literatury:**

[1] J. Mullane, B.-N. Vo, M. Adams, B.-T. Vo.: Random Finite Sets for Robot Mapping & SLAM, Springer Tracts in Advanced Robotics 72, Springer-Verlag Berlin Heidelberg, 2011.
[2] A. Doucet, N. de Freitas & N. Gordon (eds): Sequential Monte Carlo Methods in Practice. Springer-Verlag. 2001. ISBN 0-387-95146-6.
[3] S. Thrun, W. Burgard, and D. Fox. Probabilistic Robotics. MIT Press, Cambridge, MA, 2005.
[4] D. Moratuwage B.-N. Vo, and D. Wang, "Collaborative Multi-Vehicle SLAM with Moving Object Tracking", Proc. IEEE Int. Conf. Robotics & Automation, (ICRA'13), Karlsruhe, Germany, May, 2013.
[5] D. Moratuwage, B.-N. Vo, and D. Wang "A Hierarchical Approach to the Multi-Vehicle SLAM Problem", Proc. 15th Annual Conf. Information Fusion, Singapore, 2012.

**Vedoucí diplomové práce:** RNDr. Miroslav Kulich, Ph.D.

**Platnost zadání:** do konce letního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
**vedoucí katedry**

prof. Ing. Pavel Ripka, CSc.
**děkan**

V Praze dne 10. 1. 2014

## Declaration

I hereby declare that I have completed this thesis independently and that I have listed all used information sources in accordance with Methodical instruction about ethical principles in the preparation of university theses.

In Prague on..... .....

# Acknowledgements

I would like to thank my supervisor RNDr. Miroslav Kulich, Ph.D. for his support and encouragement during this project.

*Abstrakt*

Využitie náhodných konečných množín pre úlohu Simultánnej Lokalizácie a Mapovania (SLAM) je nová metóda, ktorá ukazuje sľubné výsledky minimálne v prítomnosti rušivých meraní. Cieľom tejto práce je implementovať a experimentálne overiť Multi Robotický SLAM s využitím náhodných konečných množín v 3D priestore. Simulované i reálne dáta sú spočítané tak, aby sa určila kompletná analýza presnosti i časovej závislosti navrhnutej implementácie. Nazbieraný dataset je verejne dostupný rovnako ako implementovaná knižnica pre filtráciu vo forme nezávislej C++ knižnici zabalenej do balíku Robotického Operačného Systému.

*Abstract*

The Random Finite Set approach to Bayesian Simultaneous Localization and Mapping (SLAM) is a new method which provides promising results at least in the presence of clutter measurements. The goal of this thesis is to implement and experimentally verify the Random Finite Set Multi Vehicle SLAM in 3D. Both, simulated and real life datasets are evaluated to provide complete performance and precision analysis of the proposed implementation. Moreover, the collected dataset is publicly available as well as implemented library for the Random Finite Set Filtering as well as library in a form of standalone C++ library wrapped into the Robot Operating System package.

# Contents

# List of Figures

# List of Tables

# List of Appendices

# 1 Introduction

Robotics take an important role in the real life of everyone in the recent years. Since robots are used in the industry for a long time, the domestic robots are quite new. Robotics vacuum cleaner is an great example of using mobile robots in the house. To perform even such a simple task, as an vacuum cleaning is, several problems have to be solved. Examples of the problems are collision avoiding and path planning, to find the most suitable places which robot should visit and to not destroy itself during the motion. To perform these tasks robot needs information about the surrounding environment in a form of map. Map can be created in advance or can be build simultaneously as the robot moves in the environment. This process is called *mapping* and to create an consistent map the position of the robot have to be known precisely. Information from several sensors are commonly fused to estimate the most probable position of the robot during the time. The process of the position estimating is called *localization* and if both mentioned problems are solving simultaneously the problem is called *Simultaneous Localization And Mapping*(SLAM).

If a large environment has to be explored (or cleaned) it can take a long time for single robot to do that. Multiple robots can be used to decrease the time which is necessary to solve the task. Increasing the number of robot complicates the problem of SLAM but can lead to better solution and can increase robustness because more information is fused to produce the same map. Moreover, multi vehicle scenario requires additional management of tasks to divide tasks as well as possible to use all available resources.

The goal of this thesis is to examine and implement the novel approach to the SLAM using the *Random Finite Set* (RFS) theory. RFS is mathematical concept which is suitable for the representing maps as was shown in [11]. This approach to the robotics mapping shown an promising results at least in the presence of clutter measurements. The extensions to the multi vehicle SLAM using the RFS theory is also goal of this thesis. Both, single and multi robot scenario SLAMs properties and performance are examined for the various simulated environment properties as well as real environment dataset.

Thesis is divided as follow: State of the art methods of *localization*, *mapping* and SLAM are examined at the beginning. The *Random Finite Set* is then briefly introduced and a process of derivation of the basic RFS filter called *Probability Hypothses Density* (PHD) is shown. The PHD filter is used for the SLAM problem and then extended to the multi vehicle SLAM in the next section. The implementation is then shown and explained in short. The last but one section is showing simulated results where various environment properties were simulated to evaluate the precision and performance of the implemented methods. The real environment dataset, which we collected, is then used to verify implementation in the non-simulated experiments. The work and results are then summarized in the conclusion and the feature work and extensions are proposed.

# 2 State Of The Art

The problem of the Simultaneous Localization And Mapping (SLAM) have been intensively studied in the past decades. Before the SLAM was proposed for the first time, the localization and the mapping processes have been studied separately. However, the localization and mapping are kind of 'chicken-egg problem' because to successfully estimate the location, the map have to be known and vice-versa. It was shown that uncertainties of mapping and localization are highly related and their solution leads to the unified solution called SLAM. Many methods have been proposed for both the localization, and for the mapping, and few of them will be briefly introduced in this section. These ideas will be then fused into the SLAM. Moreover, the basic algorithm for the SLAM will be shown.

## 2.1 Localization

Localization is the problem of estimating the robot position with respect to some frame of reference. The frame of reference can be known at the beginning of the localization or is estimated using some sensors such as GPS. In multi-vehicle scenario the frame of reference is often estimated based on the sensor measurements when vehicles meet each other. Localization can be based on the predicted motion only or can be supported by additional sensors. Both approaches will be considered in this section and their pros and cons will be pointed out.

### 2.1.1 Dead Reckoning

The easiest way of localization is based on the odometry information only. Let assume a wheel robot with sensors measuring the velocity of each wheel. Using this information we can estimate the robot position and orientation by integration of the measured velocity. The benefit of this method is a low price of the sensors on the other hand also the errors of the sensor are integrated. Integration means that even a small error in velocity can lead to a large error in the estimated position. Moreover, drift of the wheel is not detected by the sensors so the correction can not be included in the final estimation. Because of these disadvantages the odometry only localization is usually used only as the rough estimation which is supported by the other sensors.

### 2.1.2 Probabilistic Localization

Dead reckoning localization estimates the final position of the robot only but not the belief of that estimation. Belief of the estimated position can be propagate as well if the problem of the localization is reformulated using probabilistic theory. Let $X$ be a random variable which represents the position of the robot and $\Xi$ the state space of the robot. In practical algorithms $\Xi$ is usually reduced to the workspace of the robot and discretized

or approximated by the mixture of distributions such as Gaussian mixture model [16]. The goal of localization is to estimate belief $Bel(x)$ for each position $x \in \Xi$. The current position can be then estimated by finding maximum in the distribution:

$$x = \arg\max_{\nu \in \Xi} Bel(\nu),$$

and the belief of the estimation is given by $Bel(x)$. The belief $Bel$ is propagated based on the action $u$ (i.e. relative transformation between poses $x_{t-1}$ and $x_t$) , measured by the wheel encoders, as follows [18]:

$$Bel(x_t) = \int p(x_t|u_t, x_{t-1}) Bel(x_{t-1}) \mathrm{d}x_{t-1},$$

where $p(x_t|u_t, x_{t-1})$ is Markov density which describes the motion model. The equation above is recursive as in each time $t$ the current belief $Bel(x_t)$ depends on previous belief $Bel(x_{t-1})$ only. The initial belief $Bel(x_0)$ is usually chosen as:

$$Bel(x_0) = \begin{cases} 1 & \text{if } x_0 = \text{initial position,} \\ 0 & \text{othervise.} \end{cases}$$

An example of propagation is shown in Fig 1. It can be seen that uncertainty of the estimated position is increasing during the time. To overcome this problem, information from another sensors is usually used, which will be described in the section below.

**Motion Model** As claimed before the motion model describes how probability distribution of the current state changed based on the action input. This model takes into account uncertainty of sensor measurements as well as properties of the environment such as wheels drifts. Various motion models for different robot kinematics exist but the robot moving on a flat surface will be assumed in the rest of this thesis. The common model for planar robots is called odometric motion model [18].

### 2.1.3 Map-Based Localization

Measurements from other sensors have to be used to improve robot localization. To use an additional sensor for localization, the robot have to known information about the environment which is reliable to the sensor measurements. The collected information about the environment is called map and some of the mapping techniques will be shown in the next section. Map is expected to be created in advance for the purpose of localization in this section.

If the map exists and there is a sensor which measures the data comparable to that map we can estimate $p(z|m, x)$, where $m$ is the map and $z$ is measurement from the sensor. For example, the sensor can be a laser range finder, which measures distances to the nearest

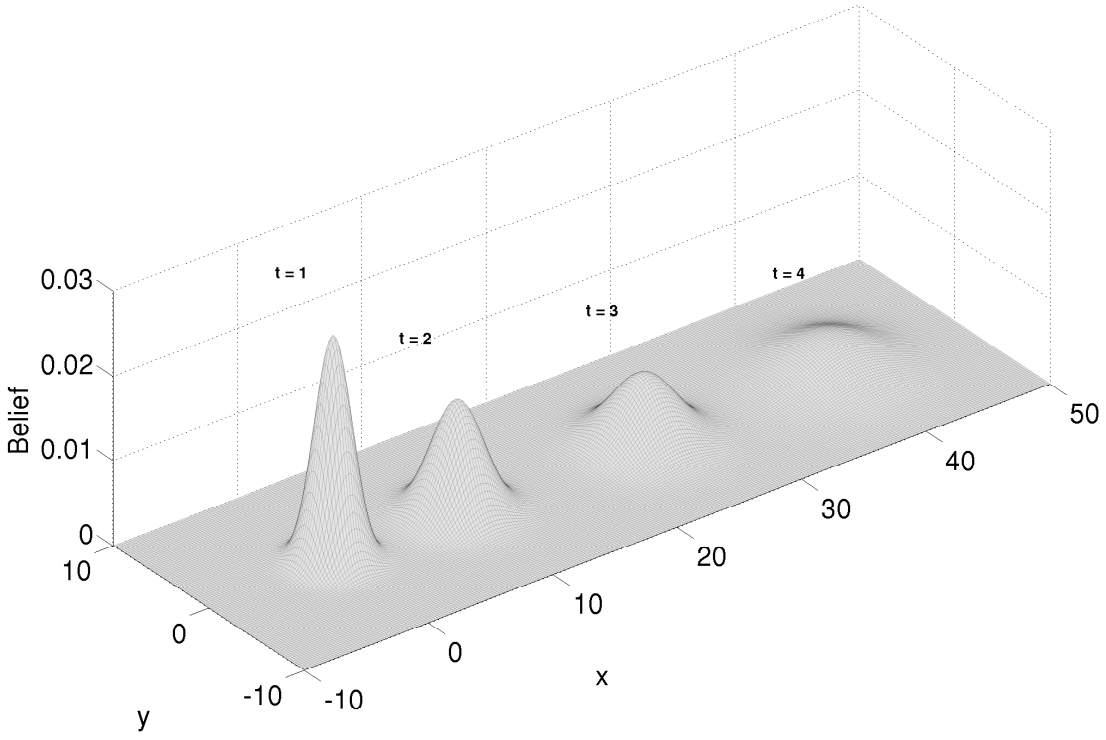



Figure 1: Propagation of the belief example

obstacle in various angles. The map can be any geometrical representation, which describes the working environment and $p(z|m,x)$ can be evaluated as the normalized average distance between measurements and the closest obstacles. The goal of the localization is to estimate $Bel(x|z,m)$ and using the Bayes rule and Markov assumption it can be computed as[18]:

$$\begin{aligned} Bel(x|z,m) &= p(x_t|z_{1:t},u_{1:t},m) \\ &= \eta p(z_t|x_t,m)\int p(x_t|u_t,x_{t-1})Bel(x_{t-1}|m)\mathrm{d}x_{t-1}, \end{aligned} \tag{1}$$

where index $1:t$ means all the measurements (or actions) from time 1 to $t$ and $\eta$ is a normalization constant ensuring that the belief over the state space integrates to 1. This equation is known as a recursive Bayes Filter.

The Bayes Filter is usually divided into two steps: prediction and update. In the prediction step, the action $u$ is integrated into belief as described in 2.1.2. No additional sensor measurement is necessary in the prediction step. In the update step the current measurement $z$ is used to improve the belief according to 1. Bayes Filter is computationally intractable in general but there exist many known approximations which mostly differ in the representation of the probability distribution.

Examples of such approximations are Histogram Filter [6], Kalman Filter [5], Particle Filter [15], among others.

**Histogram Filter** Histogram Filter hold probability distribution in a collection of bins. Each bin indicate the frequency with which data falls into each bin. Position of the robot is meant by data for the purpose of the localization.

**Kalman Filter** Kalman Filter is optimal estimator if all noise is Gaussian with zero mean. Gaussian distribution is used to represent the probability distribution of the robot position in KF. It is fast to compute but the key limitation is that it is a unimodal distribution.

**Particle Filter** Number of particles are used to represent the probability distribution in Particle Filter. Every particle hold the robot position and the weight (i.e. probability of the holden position). Particle Filter is most general filter from the listed because it can be used to represent any distribution. On the other hand it is one of the most computationally expensive filter. Particle Filter is also known as the Sequential Monte Carlo filter.

## 2.2 Mapping

Localization in a known environment was introduced in the previous section. It was claimed that information about the environment is stored in the particular representation called map. Various types of map representations exist and few of them will be shown in this section. One of the representations is a geometrical map. Geometrical maps contain information about obstacles in the environment in the form of geometrical primitives. Such a representation can be used for collision free planning as well but is usually computationally and memory expensive. Another type of map representations is a feature based map. In the feature based maps only significant points in the state space are stored which reduces the memory requirements. But they cannot be used for collision avoidance and also other problems occur such as feature detection or data association problem. Feature detection is a problem of estimating significant points in a measurement. Data association is a problem of finding correspondent points between detected features and a previously created map.

To reduce complexity of the problem the map is often assumed to be static (i.e. obstacles are not moving). The goal of mapping is to estimate posterior probability $p(m|z_{1:t}, x_{1:t})$, where $m$ is a map, $z$ are measurements and $x$ are positions of the robot. The estimation of that probability differs based on the type of the map.

### 2.2.1 Occupancy Grid Mapping

The basic type of geometrical maps is an occupancy grid. In the occupancy grid map information about the environment is discretized into the number of cells in the grid. Each grid-cell in the map holds probability whether the cell is occupied by an obstacle. Occupancy grid mapping for mobile robotics was introduced in [3] and there exist many improvements such as one described in [7].

Computation of the full posterior $p(m|z_{1:t}, x_{1:t})$ is intractable even for a small number of cells so independence between the cells is often assumed. With the independence assumption the probability can be rewritten as:

$$p(m|x_{1:t}, z_{1:t}) = \prod_i p(m^i|x_{1:t}, z_{1:t}),$$

where $m^i$ represents the $i$-th cell of the grid. The binary Bayes Filter can be used to compute $p(m^i|\cdot)$ and with the assumption of the static map the filter simplifies to:

$$p(m_t^i|z_{1:t}x_{1:t}) = \frac{p(m^i|z_t, x_t)p(z_t|x_t)p(m^i|z_{1:t-1}, x_{1:t-1})}{p(m^i)p(z_t|z_{1:t-1}, x_{1:t})}.$$

By computing the ratio $p(m^i|\cdot)/p(\neg m^i|\cdot)$, where $p(\neg m^i|\cdot)$ is $1 - p(m^i|\cdot)$ and by using the log odds notation the filter becomes highly efficient because products turns into a sum:

$$l(m^i|z_{1:t}, x_{1:t}) = \underbrace{l(m^i|z_t, x_t)}_{\text{inv. sensor model}} - \underbrace{l(m^i)}_{\text{prior}} + \underbrace{l(m^i|z_{1:t-1}, x_{1:t-1})}_{\text{recursive}},$$

where $l(\cdot)$ is log odds notation defined as

$$l(x) = \ln \frac{p(x)}{1 - p(x)}, \quad p(x) = 1 - \frac{1}{1 + \mathrm{e}^{l(x)}}.$$

**Inverse Sensor Model** The key component of Occupancy Grid Mapping is the inverse sensor model which describes a single measurement using the probability theory so the measurement noise is reflected by the model. By using the inverse sensor model the probability that a map cell is occupied is evaluated based on the robot position and the current measurements. The example of a simplified inverse sensor model for one ray of the laser range finder is shown in Figure 3.

### 2.2.2 Feature Based Mapping

The occupancy grid mapping requires to store information about every part of the state space. It usually requires lot of memory for storing such a data and thus another representations were created. One of them, feature based mapping reduces memory requirements


Figure 2: Occupancy grid map of the San Jose Tech Museum, downloaded from [18]

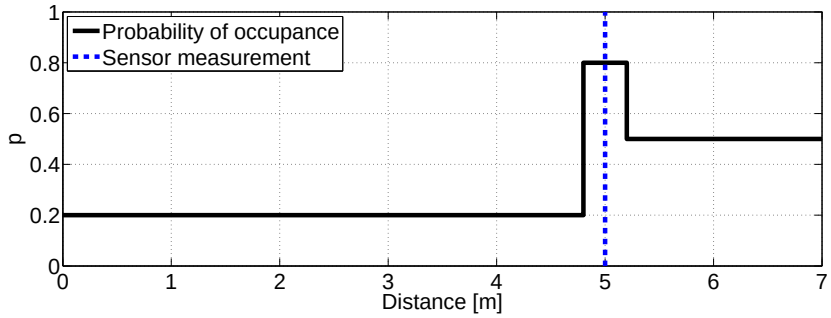



Figure 3: Simplified inverse sensor model for the one ray of the laser range finder

by storing only significant points (e.g., corner detected from the laser scan or a tree in a forest) of the space. The significant points are called landmark features and typically they are isolated. If the features are isolated, a descriptor can be used to assign current measurement with the previously stored features. A lot of methods for extracting the features exist and usually the descriptor of the feature is extracted as well during this process. Detailed information about the feature detectors can be found in [2].

The feature based mapping is introduced in this section without closer examination of feature detectors. The presence of independent isolated features is assumed and measurements will contain information about the features locations and their descriptors. Moreover it is assumed that each feature generates at maximum one measurement at the time. Extracted descriptors are used to find correspondences between the current measurement and the map, which consists of the previously detected landmarks. After correspondences are found the single target Bayes Filter can be used to incorporate the measurement into the map. The feature assignment problem depends on quality of the descriptor as well as the

consistency of the map.

The feature based mapping is a problem of estimating the posterior:

$$p(m|z_{1:t}, x_{1:t}) = p(\Theta|z_{1:t}, x_{1:t}) = \prod_i p(\theta_i|z_{1:t}, x_{1:t}),$$

where $\Theta = \{\theta_1, \ldots, \theta_k\}$ is a set of landmarks and the factorization is possible because of assumed independence between the landmarks. One measurement $z$ is assigned to every landmark $\theta$ using extracted descriptors. The Kalman Filter then can be used to update landmark position and uncertainty based on that measurement.

**Kalman Filter** A landmark in Kalman Filter (KF) is represented by the Gaussian distribution, i.e. by mean $\mu$ and covariance matrix $P$. The resulting map is in the form of the Gaussian Mixture Distribution with the equal weight of each Gaussian.

The KF was designed for linear systems only but measurement models in robotics are typically non-linear. Because of that, the extension of the KF has to be used such as Extended (EKF) or Unscented Kalman Filter (UKF) which can deal with non-linear systems as well. In fact KF and their non-linear alternatives (EKF, UKF) estimate the most probable result from noisy measurements.

Assume a static map for the simplicity. The equations which describe the system for static landmarks are:

$$\mu_t = \mu_{t-1}, \qquad z_t = h(\mu_t) + v,$$

where $h(\cdot)$ is an observation model and $v \sim N(0, R)$ is a zero mean Gaussian white noise with the covariance matrix $R$. Function $h(\cdot)$ maps the features from the state space into the observation space. The EKF is one of the many recursive Bayes Filters, divided to the prediction and update step. The prediction is straightforward because the position of the landmark remains the same in the static map:

$$\mu_{t|t-1} = \mu_{t-1}, \qquad P_{t|t-1} = P_{t-1},$$

where subindex $t|t-1$ is commonly used meaning prediction step between times $t-1$ and $t$. In the update step the measurement is incorporated into the existing landmark based on the observation model:

$$\begin{aligned} S &= HP_{t|t-1}H^T + R, & K &= P_{t|t-1}H^T S^{-1} \\ \mu_{t|t} &= \mu_{t|t-1} + K(z_t - H\mu_{t|t-1}), & P_{t|t} &= (I - KH)P_{t|t-1}, \end{aligned}$$

where $H$ is Jacobian matrix of the observation model $h(\cdot)$ and subindex $t|t$ is commonly used for the time after the update step.

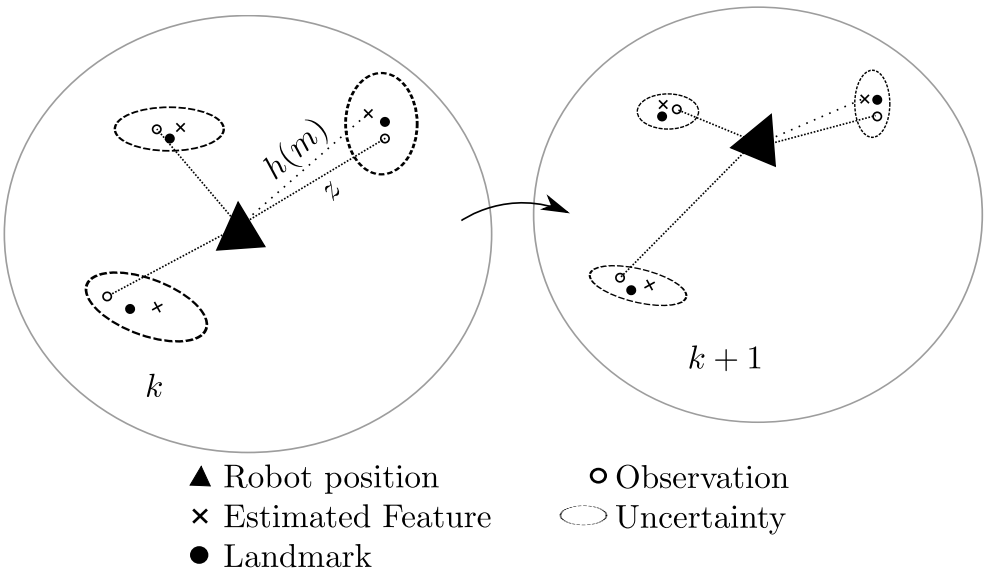



Figure 4: Example of Kalman Filter mapping

## 2.3 SLAM

The Simultaneous Localization and Mapping (SLAM) put together the ideas shown in previous sections. The problem of SLAM is to estimate the position as well as the map at the same time based on measurements:

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}),$$

where $z_{1:t}$ is a set of observations and $u_{1:t}$ is a set of actions which was sent to the robot controller. Knowledge of the observation model as well as the motion model and their parameters is assumed in this section.

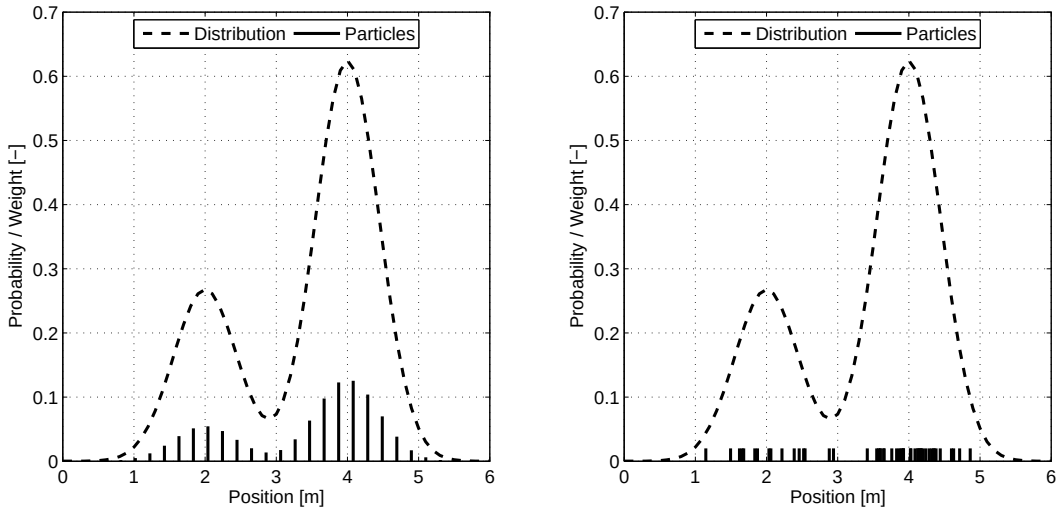
For a feature based SLAM with independent features the problem can be reformulated as:

$$p(x_{1:t}, \Theta|z_{1:t}, u_{1:t}) = p(x_{1:t}, z_{1:t}, u_{1:t})p(\Theta|x_{1:t}, z_{1:t}) = p(x_{1:t}, z_{1:t}, u_{1:t}) \prod_i p(\theta_i|z_{1:t}, x_{1:t}),$$

which is known as Rao-Blackwellize (RB) factorization. Using RB factorization the Particle Filter (PF) can be used for localization to solve the SLAM problem.

**Particle Filter** A Particle Filter represents the probability distribution by the number of independent particles. Specially for the SLAM, each particle is represented by the tuple $p_i = \{x_{1:t}, \Theta, \omega\}$, where $\omega$ represents the weight of the particle which depends on the position of the particle in the map. Each particle is propagated independently using the Bayes Filter. This approach has disadvantages of enlarging deviation of particle positions. To avoid this problem, particles are randomly re-sampled based on the discrete distribution

formed by the particles weights. After the re-sampling the most probable trajectories and maps are duplicated and particles with the lower probability are destroyed. The largest advantage of the Particle Filter is that there is no restriction about the state space and the probability density. Example of distribution approximation using particles is shown in Fig 5.




(a) Before re-sampling

(b) After re-sampling

Figure 5: Particle Filter re-sampling example

Particle filters are commonly used for the SLAM only in recent years. The reason for that is computational expensiveness of that filter. Since each particle holds its own map, mapping has to be done $n$-times, where $n$ is the number of the particles. The smallest number of the particles is not easy to determine and it depends on the quality of the motion model. If $n$ is not large enough there is possibility of loosing the correctly estimated trajectory.

# 3 Random Finite Set Theory

In the state of the art it was claimed that feature based mapping requires to solve data association problem. Data association success rate is highly related to the precision of the mapping and can lead to an inconsistent map. It is also computationally expensive and thus there was an effort to obey the association problem. One of the options how to do that is to reformulate the SLAM using Random Finite Set (RFS) theory. The Random Finite Set is mathematical representation which was used to derive first bayesian true multi-target Bayes Filter. With such a filter, all measurements can be incorporated into the map without finding the correspondences.

To use RFS for SLAM, the mathematical concepts which were used to derive multi-target BF have to be introduced. This section is divided as follows: Useful concept for RFS representation called Probability Generating Functionals (PGFL) will be shown. Using this concept, the idea for the derivation of the Probability Hypotheses Density (PHD) Filter will be introduced and then extended for the multi-sensor PHD filter. PHD Filter is one of the basic multi-target Bayes Filter approximation and can be used to solve the problem of SLAM.

## 3.1 Random Finite Set

A Random Finite Set is a finite set consisting of usually independent random variables. To describe the probability distribution of the random variables from the set, PGFL can be used. It is an extension of the Probability Generating Functions (PGF) which were introduced in [10] and can be used to describe probability distribution of one random variable. In fact PGF is used to describe probability of one landmark in the map and PGFL is used to describe all landmarks at once. Even the definition of the PGF and PGFL are different their properties are equal. In the paragraphs bellow the PGF and its properties will be shown and then extended to PGFL. The properties of the PGFL will be used to derive the PHD filter.

### 3.1.1 Probability Generating Functions

PGF is a power series representation of random variable taking only non-negative values. For a discrete random variable $X$, the PGF is the function $G_X(s)$ defined by

$$G_X(s) = \sum_{k=0}^{\infty} p_X(k) s^k,$$

where $s$ is a complex variable and $p_X(k)$ is the probability that $P(X = k)$, for $k = 0, 1, 2, \ldots$. The affect of various values of $s$ will be shown in the PGF properties section bellow.

By comparison to the definition of the expectation value $\mathbb{E}[X]$ for a discrete random variable $X$ which can take only non-negative values

$$\mathbb{E}[X] = \sum_{k=0}^{\infty} p_X(k)k,$$

it can be seen that PGF can be rewritten as

$$G_X(s) = \sum_{k=0}^{\infty} p_X(k)s^k = \mathbb{E}[s^X].$$

**Common Distributions** Many distributions exist but for the purpose of SLAM the following are the most important because they model the behavior of the landmarks and the robot. The significance of the distributions for the robotics will be shown as well as their PGFs.

- **Bernoulli Distribution** Typically the sensor is not infallible and it can miss some landmarks from the environment. It can also be a problem of a feature detector algorithm as well. To model probability whether a landmark was detected or not the Bernoulli distribution is suitable. In Bernoulli distribution there is either zero or one object, formally $P(X = k) = 0$ for all $k > 1$. PGF of the Bernoulli distribution is $G_X(s) = p_X(0) + p_X(1)s$. Because probability have to sum to 1, the Bernoulli distribution can be characterized by one number only. In robotics, using the laser range finder and the state of the art feature detector, the probability of the detections is usually around 95 %.
- **Poisson Distribution** Poisson distribution is a discrete distribution which describes some processes that occurs in nature. In the robotics the Poisson distribution can represent the number of clutter measurements of the sensor. It is often assumed that clutter measurements are independent and uniformly distributed in the field of view of the sensor and that number of clutter measurements is Poisson random variable. The Poisson distribution can be uniquely characterized by its rate $\lambda$. PGF of the Poisson distribution is $G_X(s) = \exp(\lambda(s-1))$.

**Probability Mass Function** From the PGF the probability mass function can be determined via differentiation and by substituting $s = 0$:

$$P(X = k) = G_X^{(r)}(0),$$

where $G_X^{(r)}$ means the $\mathrm{r}^{\mathrm{th}}$-order differentiation of $G_X$.

**Factorial Moments** Factorial moments are one of the statical descriptors of the probability distribution that can be determined from the PGF by differentiation as follows

$$\mu_X^{(r)} = G_X^{(r)}(1) = \mathbb{E}[X(X-1)\dots(X-r+1)],$$

where $\mu_X^{(r)}$ means r$^{\text{th}}$-order factorial moment of the variable $X$. The factorial moments will form basis for our filtering method because only the first factorial moment will be propagated to make propagation tractable.

**Sums of Independent Random Variables** For independent non-negative discrete random variables $X$ and $Y$ with generating functions $G_X(s)$ and $G_Y(s)$ the summation $X+Y$ is given by

$$G_{X+Y}(s) = \mathbb{E}[s^{X+Y}] = \mathbb{E}[s^X s^Y] = \mathbb{E}[s^X]\mathbb{E}[s^Y] = G_X(s)G_Y(s).$$

**Branching Processes** Branching processes will form the basis for the model for object tracking algorithm derived in the next section. Branching process is a Markov process that models a population in which each individual in generation $n$ produces some random number of individuals in the next generation, according to some distribution. In the language of PGF it can be expressed as $G_{n+1}(s) = G_n(G(s))$, where $G(s)$ is the PGF. In tracking application the $G(s)$ is often describing the Bernoulli distribution so the target either die or survive to the next generation.

**Joint Probability Generating Functions** To use generating functions for inference problems, the generating functions of two variables have to be considered. The joint probability generating function is defined as follow

$$G_{Z,X}(s,t) = \mathbb{E}[t^Z s^X] = \sum_{m,n=0}^{\infty} p_{Z,X}(m,n)t^m s^n.$$

From the definition of the joint PGF it is possible to determine the conditional probability mass function and factorial moments.
The conditional probability mass function:

$$p_{X|Z}(n|Z=m) = \frac{p_{Z,X}(m,n)}{p_Z(m)} = \frac{\frac{1}{n!}\frac{\partial^{m+n}}{\partial t^m \partial s^n} G_{Z,X}(t,s)\big|_{t=0,s=0}}{\frac{\partial^m}{\partial s^m} G_{Z,X}(t,1)\big|_{t=0}}.$$

And the conditional nth-order factorial moment:

$$\mu_{X|Z}^{(n)}(Z=m) = \frac{\frac{1}{n!}\frac{\partial^{m+n}}{\partial t^m \partial s^n} G_{Z,X}(t,s)\big|_{t=0,s=1}}{\frac{\partial^m}{\partial s^m} G_{Z,X}(t,1)\big|_{t=0}}.$$

### 3.1.2 Probability Generating Functionals

To extend the ideas from the previous section to deal with multi-object probability density the probability generating functionals (PGFL) are used. The PGFL for finite point processes were introduced in 1962 by Moyal [10]. The PGFL is defined as the expectation value of the symmetric function $w(x^n) = \prod_{i=1}^{n} h(x_i)$, so that

$$G(h) = \sum_{n=0}^{\infty} \int \left( \prod_{i=1}^{n} h(x_i) \right) p(x_1, \ldots, x_n) \mathrm{d}x_1 \ldots \mathrm{d}x_n.$$

Similarly the joint PGFL:

$$G(h, g) = \sum_{n,m=0}^{\infty} \iint \left( \prod_{i=1}^{n} h(x_i) \right) \left( \prod_{j=1}^{n} g(y_j) \right) p(x_1, \ldots, x_n, y_1, \ldots, y_m) \mathrm{d}x_1 \ldots \mathrm{d}x_n \mathrm{d}y_1 \ldots \mathrm{d}y_m.$$

**PGFL of Common Point Processes** A point process is a mathematical description of a process that generates points in time according to defined probability distribution. Point is discrete event that occurs in the time and points are isolated.

- **Bernoulli point process**

$$G(h) = p_0 + \int p_1(x) h(x) \mathrm{d}x.$$

- **Poisson point process** In Poisson point process the number of points is distributed based on Poisson random variable and that points are distributed in space according to distribution $s(x)$.

$$G(h) = \exp(\lambda( \int h(x) s(x) \mathrm{d}x - 1)).$$

- **IID cluster process**

$$G(h) = \sum_{n=0}^{\infty} \int p(n) \left( h\left(x\right) s\left(x\right) \mathrm{d}x \right)^n,$$

  where $p(n)$ is probability mass function describing number of points and each object is independent and identically distributed according to $s(x)$.

**Functional Derivative** In PGF the ordinary derivative was used to obtain the mass function and factorial moments from PGF representation. For PGFL the ordinary derivative have to be replaced by the functional derivative which is defined as follow. Functional derivative of functional $f(h)$ in direction $\varphi(x)$ is given by

$$\delta f(h; \varphi) = \lim_{\varepsilon \to 0} \frac{f(h + \varepsilon\varphi) - f(h)}{\varepsilon}.$$

Usually, differentiation is taken with respect to some arbitrary function $\varphi(x)$ and then set $\varphi(x) = \delta_y(x)$, where $\delta_y(x)$ is dirac impulse in the direction $y$.

**Probability Mass Function and Factorial Moments** The k[th]-order density functions and the k[th]-order factorial moment can be determined from the PGFL as follows:

$$p(x_1, \ldots, x_k) = \delta^k G(h; \delta_{x_1}, \ldots, \delta_{x_k})\big|_{h=0},$$

$$m(x_1, \ldots, x_k) = \delta^k G(h; \delta_{x_1}, \ldots, \delta_{x_k})\big|_{h=1}.$$

**Bayesian Estimation** In Bayessian estimation, the goal is to find conditional probability $p(x_1, \ldots, x_n|z_1, \ldots, z_m)$. Let PGFL of that conditional probability be $G(h|z_1, \ldots, z_m)$, it can be estimated from the joint PGFL $F(g, h)$ as follow

$$G(h|z_1, \ldots, z_m)\big|_{h=0} = \frac{\delta^m F(g, h; \varphi_{z_1}, \ldots, \varphi_{z_m})\big|_{g=0}}{\delta^m F(g, 1; \varphi_{z_1}, \ldots, \varphi_{z_m})\big|_{g=0}}.$$

From the conditional PGFL the probability mass function can be computed via differentiation

$$p(x_1, \ldots, x_n|z_1, \ldots, z_m) = \delta^n G(h|z_1, \ldots, z_m; \varphi_{x_1}, \ldots, \varphi_{x_m})\big|_{h=0}.$$

The mean value can be determined from the conditional PGFL with

$$D(x) = \delta G(h|z_1, \ldots, z_m; \varphi_x)\big|_{h=1}.$$

The mean $D(x)$ is known as the Probability Hypothesis Density or PHD.

## 3.2 Probability Hypotheses Density Filter

Probability Hypotheses Density (PHD) filter is one of the basic filters in the RFS theory. The idea behind the PHD filter is to propagate only first factorial moment. Like most of the Bayesian filters the PHD filter is recursive, divided to the prediction and update step:

$$\cdots \longrightarrow G_{\Phi_{k-1|k-1}}(h) \overset{\text{predictor}}{\longrightarrow} G_{\Phi_{k|k-1}}(h) \overset{\text{corrector(udpate)}}{\longrightarrow} G_{\Phi_{k|k}}(h) \longrightarrow \cdots$$

At the beginning of time step $k$ the prior information on the multi target configuration (target numbers and spatial distribution) is described by the point process $\Phi_{k-1|k-1}$. This point process has PGFL $G_{\Phi_{k-1|k-1}}(h)$ and first factorial moment $D_{k-1|k-1}(x)$. Only the first moment is known, propagated from the past observations.

### 3.2.1 Prediction Step

Notation which represents prediction: $G_{\Phi_{k-1|k-1}}(h) \to G_{\Phi_{k|k-1}}(h)$.

**Modeling Assumptions**

- Motion Process
  A target in the state $x$ dies with probability $1 - p_{s,k}(x)$. If it survives it moves to some state $y$ with probability $f_k(y|x)$. PGFL representation of the motion process is thus
  $$G_{f,k}(h|x) = 1 - p_{s,k}(x) + p_{s,k}(x) \int h(y) f_k(y|x) dy$$
- Birth Process
  Newborn targets are distributed in the state space according to a Poisson distribution, with intensity $\lambda_{\gamma,k}$. Birth process represented by PGFL:
  $$G_{\gamma,k}(h) = \exp\left(\lambda_{\gamma,k}\left(\int h(x)s(x)dx - 1\right)\right)$$
- Predicted Process
  The target configuration after the prediction step is composed of the newborn targets and the prior targets which survived and evolved to a new state. PGFL representation:
  $$G_{\Phi_{k|k-1}}(h) = G_{\gamma,k}(h) G_{\Phi_{k-1|k-1}}(G_{f,k}(h|\cdot))$$

**Filtering** The goal is to propagate the first moment

$$D_{k|k-1}(x) = \delta G_{\Phi_{k|k-1}}(h, \delta x)\big|_{h=1}.$$

A direct calculation shows [8] that the prediction equation is

$$D_{k|k-1}(x) = b_k(x) + \int f_k(x|\nu) p_{s,k}(\nu) D_{k-1|k-1}(\nu) \mathrm{d}\nu,$$

where $b(x)$ denotes the intensity of a new target Poison point process at $x$.

### 3.2.2 Update Step

Notation which represents update: $G_{\Phi_{k|k-1}}(h) \to G_{\Phi_{k|k}}(h)$.

**Modeling Assumptions**

- Observation Process
  A target in state $x$ is detected with probability $p_{d,k}(x)$. If it is detected, id produce an observation $z$ with probability $g_k(z|x)$. PGFL of the observation process is
  $$G_{L,k}(g|x) = 1 - p_{d,k}(x) + p_{d,k}(x) \int g(z) f_k(z|x) \mathrm{d}z.$$

- Clutter Process
  False alarms are distributed in observation space according to Poisson distribution with intensity $\lambda_{c.k}$. Clutter process PGFL is

$$G_{c,k}(g) = \exp\left(\lambda_{c.k}\left(\int g(z)s_{c,k}(z)\mathrm{d}z - 1\right)\right)$$

- Predicted Process
  The predicted process is assumed Poisson, with mean value equal to predicted number of targets.

$$G_{\Phi_{k|k-1}}(h) = \exp\left(\mu_{\Phi_{k|k-1}}\left(\int h(x)s_{\Phi_{k|k-1}(x)}\mathrm{d}x - 1\right)\right)$$

- Updated Process
  The joint configuration of targets and observation is composed of the false alarms and joint configuration of predicted targets and associated observations. Updated process represented by PGFL:

$$F(g,h) = G_{c,k}(g)G_{\Phi_{k|k-1}}(hG_{L,k}(g|\cdot)).$$

**Filtering** Using Bayes Rule and assuming that the sensor produced measurements $z_1, \ldots, z_m$, the goal is to propagate first moment

$$D_{k|k}(x|\{z_1,\ldots,z_n\}) = \frac{\delta^{n+1}F(g,h,\delta_{z_1},\ldots,\delta_{z_n},\delta_x)\big|_{g=0,h=1}}{\delta^n F(g,1,\delta_{z_1},\ldots,\delta_{z_n})\big|_{g=0}}$$

It can be shown that update equation will become

$$D(x|z_1,\ldots,z_n) = D_{k|k-1}(x)(1-p_{d,k}(x)) + \sum_{z\in Z}\frac{p_{d,k}(x)g(z|x)D_{k|k-1}(x)}{\lambda_{c,k}c(z) + \int p_{d,k}(\nu)g(z|\nu)D_{k|k-1}(\nu)\mathrm{d}\nu},$$

where $D_{k|k-1}(x) = \mu_{\Phi_{k|k-1}}s_{\Phi_{k|k-1}}(x)$.

## 3.3 Multi-sensor PHD filter

The core PHD filter described in the previous section presumes a single sensor. The iterated-corrector approximation can be used to address multi-sensor scenario but it has a peculiarity: changing the order of the sensors produces different PHDs. In [9] it was said that iterated-corrector solution is enough for the practical application, even if it is not satisfactory from a theoretical point of view. In the same paper Mahler derived the two-sensor PHD filter and showed that even in the two-sensor scenario there is large differences in performance and equations becomes intractable.

### 3.3.1 Prediction

Since prediction step does not depend on sensor measurements there will be no difference from the single-sensor scenario.

### 3.3.2 Update

The variables which were used for a single sensor PHD ($\lambda$, $p_D$, $f_k$) stay the same and the number above the variable will denotes the id of the sensor. The update equation will be slightly different with respect to the single sensor scenario, but the assumptions which were assumed in the previous section are the same. Additional assumption will be that the sensors deliver their measurement sets $\overset{1}{Z}_k$ and $\overset{2}{Z}_k$ at the same time:

$$Z_k = \overset{1}{Z}_k \cup \overset{2}{Z}_k$$

The corrector equation for the two-sensor PHD filter is

$$D_{k|k}(x|Z_k) = L_{Z_k}(x|Z_k) D_{k|k-1}(x|Z_k)$$

where the two-sensor likelihood function is

$$L_{Z_k}(x|Z_k) = (1 - \overset{1}{p}_d(x))(1 - \overset{2}{p}_d(x))$$

if $Z_k = \emptyset$ and, if otherwise,

$$L_{Z_k}(x|Z_k) = (1 - \overset{1}{p}_d(x))(1 - \overset{2}{p}_d(x)) + \sum_{\mathcal{P} \boxminus_2 Z_k} \omega_{\mathcal{P}} \sum_{W \in \mathcal{P}} p_W(x)$$

where the summation is taken over all so called 'binary partitions' $\mathcal{P}$ of $Z_k$. Here,

$$p_W(x) = \begin{cases} \frac{\overset{1}{p}_d(x) \underset{\overset{1}{z}}{\overset{1}{l}_1}(x)(1-\overset{2}{p}_d(x))}{1+D_{k|k-1}[\overset{1}{p}_d \underset{\overset{1}{z}}{\overset{1}{l}_1}(1-\overset{2}{p}_d)]} & \text{if } W = \{\overset{1}{z}\} \\ \frac{(1-\overset{1}{p}_d(x))\overset{2}{p}_d(x) \underset{\overset{2}{z}}{\overset{2}{l}_2}(x)}{1+D_{k|k-1}[(1-\overset{1}{p}_d)\overset{2}{p}_d \underset{\overset{2}{z}}{\overset{2}{l}_2}]} & \text{if } W = \{\overset{2}{z}\} \\ \frac{\overset{1}{p}_d(x) \underset{\overset{1}{z}}{\overset{1}{l}_1}(x)\overset{2}{p}_d(x) \underset{\overset{2}{z}}{\overset{2}{l}_2}(x)}{D_{k|k-1}[\overset{1}{p}_d \underset{\overset{1}{z}}{\overset{1}{l}_1} \overset{2}{p}_d \underset{\overset{2}{z}}{\overset{2}{l}_2}]} & \text{if } W = \{\overset{1}{z}, \overset{2}{z}\} \end{cases}$$

$$\underset{\overset{1}{z}}{\overset{1}{l}_1}(x) = \frac{\overset{1}{L}_{\overset{1}{z}}(x)}{\overset{1}{\lambda}_k \overset{1}{c}_k(\overset{1}{z})}, \quad \underset{\overset{2}{z}}{\overset{2}{l}_2}(x) = \frac{\overset{2}{L}_{\overset{2}{z}}(x)}{\overset{2}{\lambda}_k \overset{2}{c}_k(\overset{2}{z})}$$

$$\omega_{\mathcal{P}} = \frac{\prod\limits_{W\in\mathcal{P}} d_W}{\sum\limits_{\mathcal{Q}\boxminus_2 Z_k} \prod\limits_{W\in\mathcal{Q}} d_W}$$

$$d_W = \begin{cases} 1 + D_{k|k-1}[\overset{1}{p}_d \overset{1}{l}_{\underset{z}{1}}(1-\overset{2}{p}_d)] & \text{if } W = \{\overset{1}{z}\} \\ 1 + D_{k|k-1}[(1-\overset{1}{p}_d)\overset{2}{p}_d \overset{2}{l}_{\underset{z}{2}}] & \text{if } W = \{\overset{2}{z}\} \\ D_{k|k-1}[\overset{1}{p}_d \overset{1}{l}_{\underset{z}{1}} \overset{2}{p}_d \overset{2}{l}_{\underset{z}{2}}] & \text{if } W = \{\overset{1}{z}, \overset{2}{z}\} \end{cases}$$

A binary partition of $Z_k$ was defined in [9] as follows. An arbitrary partition of $Z_k$ is a subset $\mathcal{P}$ of the class of all subsets of $Z_k$, excluding the null set, such that the union of all of the elements of $\mathcal{P}$ is $Z_k$. Every $W \in \mathcal{P}$ must have one of the following three forms:

$$W = \{\overset{1}{z}\}, \qquad W = \{\overset{2}{z}\}, \qquad W = \{\overset{1}{z}, \overset{2}{z}\}.$$

The notation $\mathcal{P} \boxminus_2 Z_k$ is a shorthand for $\mathcal{P}$ partitions $Z_k$ into binary cells $W$.

The complexity of the proposed multi-sensor PHD filter is high even for two sensors. Because of that only the iterated solution to the multi-sensor PHD filter is used in our implementation. The derivation was shown here just to preserve completeness of the PHD filter solutions.

# 4 Random Finite Set SLAM

RFS theory was introduced and PHD filter was derived both for the single and multiple sensors scenario in the previous section. It was theoretical derivation and no impact was given to the implementation, which will be fixed in this section. In the section 2 the Particle Filter (PF) was introduced and it will be used for the RFS SLAM too. For PF it is typical that each particle holds its own map and the position of the robot as well. Particles are independent and since the pose is known for each particle only the mapping has to be done. It was claimed that mapping is a problem of incorporating current measurements into a known map and to avoid data association the multi-object Bayes Filter has to be used. The PHD filter will be used for the robotics mapping in this section.

## 4.1 PHD Mapping

Two implementations of the PHD filter were derived [19, 12], Particle Filter implementation and Gaussian Mixture Model (GMM) implementation. Both implementations differ only in a way they represent the density. In particle filter the density is represented by a set of independent particles and in GMM it is represented by the weighted sum of independent Gaussian distributions. Since the GMM implementation is computationally faster the main focus will be on it in this thesis.

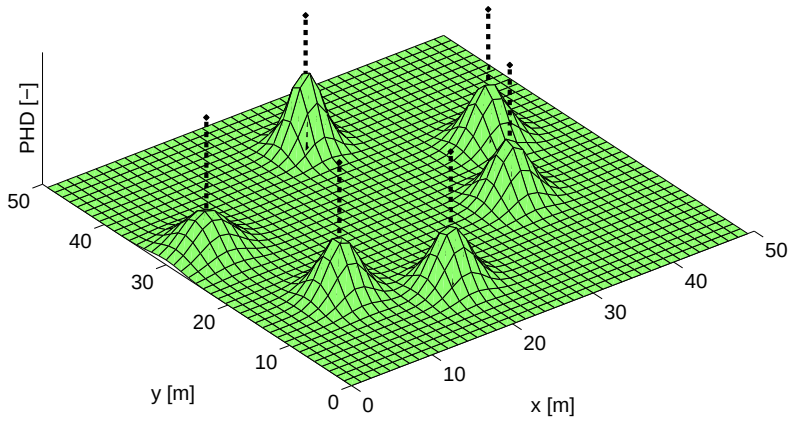



Figure 6: Map represented by the PHD and the estimated landmarks

In PHD mapping the map is represented by the intensity function called PHD. The example of this representation is shown in Figure 6. One of the fundamental properties

of the PHD is that the sum over the state space is equal to the expected number of landmarks in the map. It is the only difference to the probability density because the probability density sums to 1. The landmarks can be extracted from the map by finding $N$ maximums, which in GMM implementation means to find $N$ maximum weighted Gaussians where $N$ is the expected number of the landmarks.

### 4.1.1 Pseudo Static Map

To simplify the problem of mapping, a pseudo static map will be assumed in which the features are static but the map is enlarging in time as the robot explores a new unobserved space of the environment. This map describes behavior of the robot mapping with a limited filed of view (FoV) of the sensor. Formally, let the map be represented by the finite set $\mathcal{M}_{k-1}$ which was observed by an on board sensor with limited FoV

$$\mathcal{M}_{k-1} = FoV(X_{0:k-1}) = FoV(X_0) \cup \ldots \cup FoV(X_{k-1}).$$

### 4.1.2 GMM Approximation

In GMM implementation the prior intensity $D_{k-1}$ is approximated by the weighted sum of independent Gaussians

$$D_{k-1}(m|X_{k-1}) = \sum_{j=1}^{J_{k-1}} \omega_{k-1}^{(j)} \mathcal{N}(m; \mu_{k-1}^{(j)}, P_{k-1}^{(j)}),$$

where $\mathcal{N}(x; \mu, P)$ is an acronym for the probability of state $x$ of Gaussian distribution with the mean value $\mu$ and covariance matrix $P$. The new observed features entering the FoV are also represented by the Gaussian Mixture Model:

$$b_k(m|Z_{k-1}, X_{k-1}) = \sum_{j=1}^{J_{b,k}} \omega_{b,k}^{(j)} \mathcal{N}(m; \mu_{b,k}^{(j)}, P_{b,k}^{(j)}),$$

where $J_{b,k}$ defines the number of the new features intensity at time $k$. The individual Gaussians in $b_k$ are constructed from the observed features $Z_k$ as follow:

$$\omega_{b,k}^{(j)} = 0.01, \qquad \mu_{b,k}^{(j)} = h^{-1}(z_k^{(j)}, X_k),$$
$$P_{b,k}^{(j)} = h'(\mu^{(j)}, X_k)\, R\, h'(\mu^{(j)}, X_k)^T,$$

where $h^{-1}$ is the inverse measurement model, $R$ is the measurement noise covariance matrix and $h'$ is the Jacobian of the measurement model function. Because of static map assumption the predicted intensity will consist of previously created Gaussians and the new observed Gaussians

$$D_{k|k-1} = D_{k-1} + b_k = \sum_{j=1}^{J_{k|k-1}} \omega_{k|k-1}^{(j)} \mathcal{N}(m; \mu_{k|k-1}^{(j)}, P_{k|k-1}^{(j)}),$$

where $J_{k|k-1}$ is the predicted number of Gaussians which is the sum of the number of prior and new observed Gaussians.

### 4.1.3 GMM Update

Gaussian Mixture Model update equation recomputes the intensity function based on the current measurements $Z_k$. In GMM update equation changes to form:

$$D_{k|k}(m|X_k) = D_{k|k-1}(m|X_k)\left[1 - P_D(m|X_k) + \sum_{z\in Z_k}\sum_{j=1}^{J_{k|k-1}} D_{g,k}^{(j)}(z, m|X_k)\right], \tag{2}$$

where individual components of that equation are computed as follow:

$$D_{g,k}^{(j)}(z, m|X_k) = \omega_k^{(j)}(z|X_k)\mathcal{N}(m; \mu_{k|k}^{(j)}, P_{k|k}^{(j)}),$$

$$\omega_k^{(j)}(z|X_k) = \frac{P_D(m|X_k)\omega_{k|k-1}^{(j)} q^{(j)}(z, X_k)}{c_k(z) + \sum_{i=1}^{J_{k|k-1}} P_D(m|X_k)\omega_{k|k-1}^{(j)} q^{(i)}(z, X_k)},$$

where $q^{(i)}(z, X_k)$ is Gaussian with values $\mathcal{N}(z; h_k(\mu_{k|k-1}^{(i)}), S_k^{(i)})$. The components $\mu_{k|k}^{(j)}, P_{k|k}^{(j)}$ and $S_k^{(i)}$ are obtained from the Extended Kalman Filter:

$$\begin{aligned} S_k^{(i)} &= H_k P_{k|k-1}^{(i)} H_k^T + R_k, & K_k^{(i)} &= P_{k|k-1} H_k^T [S_k^{(i)}]^{-1}, \\ \mu_{k|k}^{(i)} &= \mu_{k|k-1} + K_k^{(i)}(z_k - h_k(\mu_{k|k-1}^{(i)})), & P_{k|k}^{(i)} &= (I - K_k^{(i)} H_k) P_{k|k-1}^{(i)}, \end{aligned}$$

with $H_k$ being the Jacobian of the measurement model $h_k(\cdot)$. As stated in the previous section, the clutter is assumed Poisson distributed in number of features and uniformly spaced in the sensor field of view (FoV). Therefore the clutter is given by:

$$c_k(z) = \lambda_c V U(z),$$

where $\lambda_c$ is the clutter density per measurement, $V$ is the volume of the FoV and $U(z)$ denotes a uniform distribution on the FoV.

Probability of detection depends on the robot position and on the sensor field of view. It is evaluated as follow:

$$P_D(m, X_k) = \begin{cases} P_D & \text{if is in FoV} \\ 0 & \text{otherwise} \end{cases},$$

where $P_D$ is constant which has to be tunned based on the sensor and feature detector properties and based on the mapped environment.

### 4.1.4 Gaussians Management

In the update equation (2) it can be seen that $J_{k|k-1}$ new Gaussians are created for each sensor measurement in each time step. This property of the GMM implementation leads to computational instability of the mapping. To avoid the computational intractability additional approximation has to be done which consists of three steps:

- Gaussians with weight lower than the defined threshold are removed.
- Gaussians with Mahalanobis distance smaller then defined threshold are merged.
- The smallest Gaussians are removed to preserve maximum number of stored Gaussians.

The approximation requires to tune three parameters with which the balance between performance and precision can be found.

## 4.2 PHD SLAM

PHD SLAM uses a Particle Filter to represent the number of possible maps with associated robot positions or trajectories. The following process is used and repeated for each SLAM input which consists of robot motion measurement and sensor measurement. Each particle in Particle Filter is propagated in the state space based on the motion model and robot motion measurement. A propagated particle then incorporates the sensor measurements into its own map. To incorporate measurements the PHD Filter is used. After mapping is done particles are weighted and resampled. Next input is processed after resampling is done.

### 4.2.1 Particle Weighting

Particle weighting is the process of estimating probability of the current measurement based on the created map and the robot position. Two weighting strategies [11] are commonly used for the RFS SLAM which differ in the expected number of features in the map. The 'multiple feature map' weighting strategy exists but is computationally more expensive and not necessary for our implementation.

**Empty Map Strategy** In the empty map strategy the the weight is computed as:

$$p(Z|M,X) = \exp\left(m_k - m_{k|k-1} - c_k(z|X)\right) \prod_{z \in Z} c_k,$$

where $m_k$ is the estimated number of features in the map after the update step and $m_{k|k-1}$ is estimated number of features in prior map. Based on the properties of the PHD the number of features can be computed as the integral over the state space:

$$m_k = \int_{x \in S} D_k(x).$$

Even the equation for weighting in the empty map strategy does not contain the measurement likelihood, the current and past measurements are considered because they were incorporated into the probability hypotheses density $D_k$ and $D_{k|k-1}$.

**Single Feature Strategy** Even the empty map strategy is correct from mathematical point of view the number of features in weighting does matter in a practical algorithm. The reason why mathematical correctness does not imply the practical correctness is the presence of approximation. Because of that the single feature strategy is explained here where one feature is chosen from the map and used for particle weight computation. Any arbitrary strategy for picking the feature can be used, for example the feature represented by the highest weighted Gaussian or the lowest uncertainty Gaussian, etc. The weight is computed as:

$$p(Z|M,X) = \frac{1}{\Gamma}\left[\left(\left(1 - P_D(m|X)\right)\kappa^Z + P_D(m|X)\sum_{z\in Z} g(z|m,X)\kappa^{Z-\{z\}}\right) D_{k|k-1}(m|X)\right],$$

where $m$ is the selected feature, $\kappa^Z$ is computed from the clutter model as $\prod_{z\in Z} c_k(z|X)$ and $\Gamma$ stands for $\exp\left(m_k - m_{k|k-1} - c_k\right) D_k(m|X)$. For the purpose of SLAM it is important to chose the feature which is in the FoV of the sensor. Otherwise all measurements are compared to the invisible feature which will lead to the equal weight for each particle.

### 4.2.2 Trajectory Estimation

Each particle stores the trajectory from the start to the current position. To estimate the final robot trajectory two approaches can be used: maximum a posteriori and expected a posteriori. The first one estimates the position of the robot in time $k$ from the maximum weighted particle only:

$$x_k^{\text{MAP}} = \underset{x_k}{\text{argmax}}\left(p(z_k|x_k)\right),$$

where maximum is found over all particles. The expected a posteriori estimate the position for each time $k$ as a weighted average position:

$$x_k^{\text{EAP}} = \frac{1}{N}\sum_{i=1}^{N} x_k^i p(z_k|x_k^i),$$

where $x_k^i$ stands for $i$-th particle and $N$ is number of particles.

## 4.3 Multi Vehicle SLAM

Two approaches to extension of the single robot SLAM to the multi vehicle SLAM (MVSLAM) are described in this section. The first one expects knowledge of initial correspondences where start positions of vehicles are known while the second one deals with the unknown initial correspondences. Both approaches use the RFS for map representation and multi sensor PHD filter for the measurements incorporation. The iterative multi sensor PHD filter is used because of performance (i.e. single sensor PHD filter is called $N$-times on the same map where $N$ is the number of vehicles). Similarly to the single robot SLAM, Particle Filter is used to propagate positions of each vehicle in time.

### 4.3.1 Known Initial Correspondences

Assume knowledge of initial correspondences of robot poses first (Fig. 7). The correspondences can be measured by the operator or estimated automatically by an external sensor. Note that knowledge of relative initial positions between robots is enough for the purpose of the MVSLAM so determination of the positions with respect to some global coordinate system (GPS for example) is not necessary. If $N$ robots are used for MVSLAM each particle in the particle filter in time $k$ consists of the touple:

$$\left(D_k, \ \overset{1}{x}_k, \ \overset{2}{x}_k, \ \ldots, \ \overset{N}{x}_k\right),$$

where $D_k$ is a map intensity function and $\overset{i}{x}_k$ represents a position of the $i$-th robot in the map.

Each position is propagated independently using odometric motion model. Because the map is represented by the Random Finite Set, measurements can be incorporated into the map iteratively. It reduces the complexity of mapping significantly with respect to multi-sensor PHD Filter while producing the similar result. The particle weighting for multi vehicle SLAM differs from single robot SLAM in the way the probability is evaluated. For an empty map strategy it is the same but for a single feature strategy individual probabilities are multiplied:

$$p(\overset{1}{Z}, \ldots, \overset{N}{Z}|M, \overset{1}{x}, \ldots, \overset{N}{x}) = \prod_i p(\overset{i}{Z}|M, \overset{i}{x}),$$

where $p(\overset{i}{Z}|M, \overset{i}{x})$ is computed as in the single robot scenario and such a factorization is possible because of assumption of independence between vehicles.

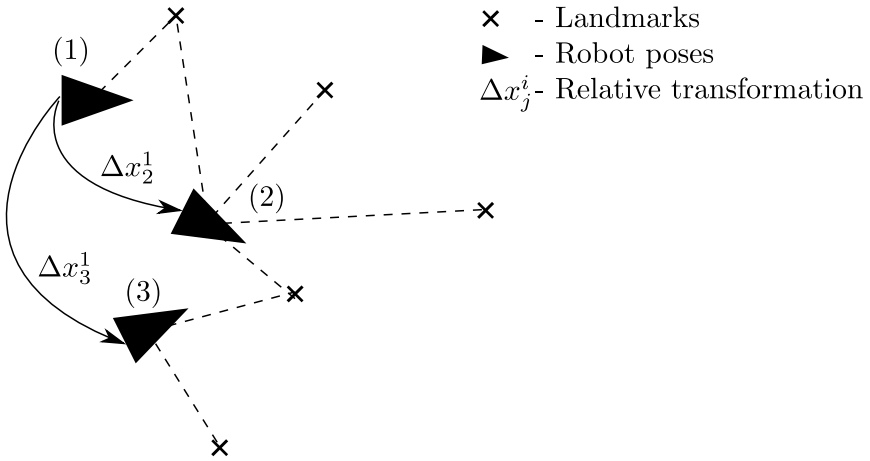



Figure 7: Initial step of the MVSLAM with known initial correspondences

### 4.3.2 Unknown Initial Correspondences

Knowledge of initial correspondences is not available in many real life applications. Thus performing the MVSLAM with unknown initial correspondences is more related with real life experiments but it is also more challenging. The approach described in [21] is adopted for the purpose of MVSLAM. Several robots perform SLAM independently and the maps are merged when robots meet each other, in that approach. The ability to estimate relative robots positions when the robots meet is assumed. Moreover, the measurements and actions which were used to build the map have to be stored to perform data fusion.

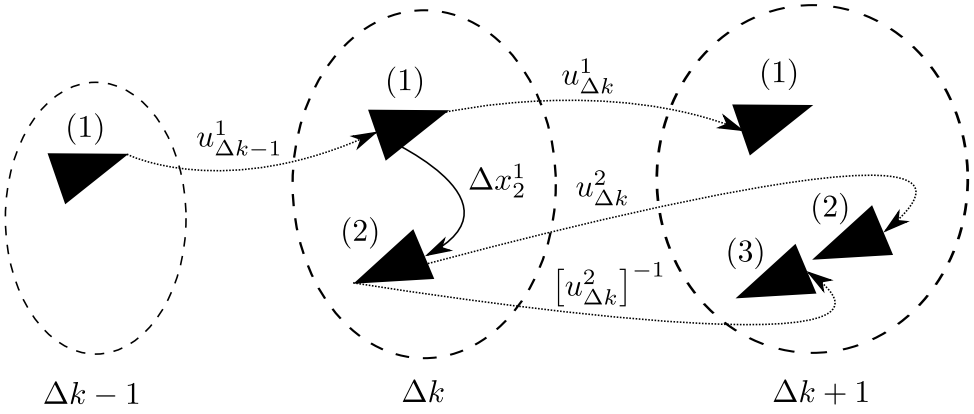
**Two Robots Scenario** Simplified, two robots scenario (Fig. 8) is used to briefly introduce the used concept. The scenario can be extended to any number of robots if necessary. Formally, scenario consists of timestamped positions $\overset{i}{x}_k$ and observations $\overset{i}{z}_k$. Let assume robots meet at time $\Delta k$ and the robot $\overset{1}{r}$ estimates a relative position $\Delta x_2^1$ of the second robot $\overset{2}{r}$. The mapping process is described from the view of the first robot only but the second robot can make its own map simultaneously if real time output of the SLAM is required for other purposes (such as navigation, planning, etc. ). The first robot is performing single robot SLAM using PHD filter and thus producing map $D_k$ and position $\overset{1}{x}_k$. In time $\Delta k$ the robots meet and relative transformation is used to estimate position of the second robot in the map $D_{\Delta k}$:

$$\overset{2}{x}_{\Delta k} = \overset{1}{x}_{\Delta k} \cdot \Delta x_2^1.$$

From that point, the multi vehicle SLAM is used to incorporate measurements from both robots into a single map. To incorporate second robot measurements from time before $\Delta k$, a virtual robot is used. The virtual robot travels back in the time to incorporate all past measurements and thus improving the map estimation. It is constructed based on the second robot measurements and inverted actions as:

$$\overset{3}{u}_{k+i} = \left[\overset{2}{u}_{k-i}\right]^{-1}, \quad \overset{3}{z}_{k+i} = \overset{2}{z}_{k-i} \qquad i \in \{1, \ldots, \Delta k\}.$$

The advantage of this iterative solution for the past measurements incorporation is that the robots do not lag when they meet. The alternative solution, where all past measurements are incorporated when robots meet can lead to loosing the correct estimation if incorporation is not fast enough and there is not enough particles representing the probability of positions.




| | | | |
|---|---|---|---|
| $\Delta x$ | Relative transformation | ►(1,2) | Robot pose |
| $u$ | Robot input | ► (3) | Virtual robot pose |
| $\Delta k$ | Meeting time | | |

Figure 8: MVSLAM with unknown initial correspondences

# 5 Implementation

Proposed methods are implemented in the C++ programming language as a template libraries. Test driven development was used for library development using Google's framework for writing C++ test called *GTest*. Developed libraries were used to implement MVS-LAM into the *Robot Operating System* (ROS). Note that even the implementation was developed under the ROS, the libraries them self are ROS independent and thus usable in any other codes. The only prerequisites are presence of *Eigen* library, which was used to perform algebraic computation, and *BOOST* libraries of version at least 1.49. Both required libraries are open source and available for Linux, Mac and Windows as well.

## 5.1 Robot Operating System

The Robot Operating System was designed few years ago and during that years it became standard for the robotic applications. ROS is designed for Linux operating systems and only officially supported distribution is Ubuntu. Because of *BOOST* dependency in our implementation, the libraries are compatible only with the following version of ROS:

- *ROS Indigo Igloo*, all Indigo supported Ubuntu distributions (13.10, 14.04, so far),
- *ROS Hydro Medusa*, for Ubuntu 13.04 (Ubuntu 12.04 has older version of *BOOST*).

Note that any older version of ROS/Ubuntu can be used if the new *BOOST* version is installed at least locally.

## 5.2 Test Framework

Test driven development is one of the most popular development methods in the recent years. Developing methods using the test driven development lead to better organized and more trust-able codes. The main idea is to design tests before implementation of the method is created. Tests are small methods which compares expected and real return value of the function for various scenarios. For example testing of function for field of view (FoV) of the sensor can consists of two scenarios: testing the landmarks in the FoV, where expected return value is true and testing the landmarks outside the FoV. Many frameworks for code testing exists but *GTest* framework is used because it is natively supported by the build system inside the ROS.

It is common procedure to leave the code optimization to the end of the development. If the test were written correctly the correctness of optimization can be tested in the one command. The same tests can be used if the new version of any used library is released and thus logical mismatch in the code is easily detectable. This is the main advantage of the using test driven development because it lead to stable and not bugged code for a long period of time.

## 5.3 Libraries

Several libraries were implemented to simplify implementation of the various SLAM scenarios. All libraries are briefly introduced in this section and the library hierarchy is shown in Fig. 9. Detailed documentation can be extracted from the source codes using *Doxygen* API documentation tool.

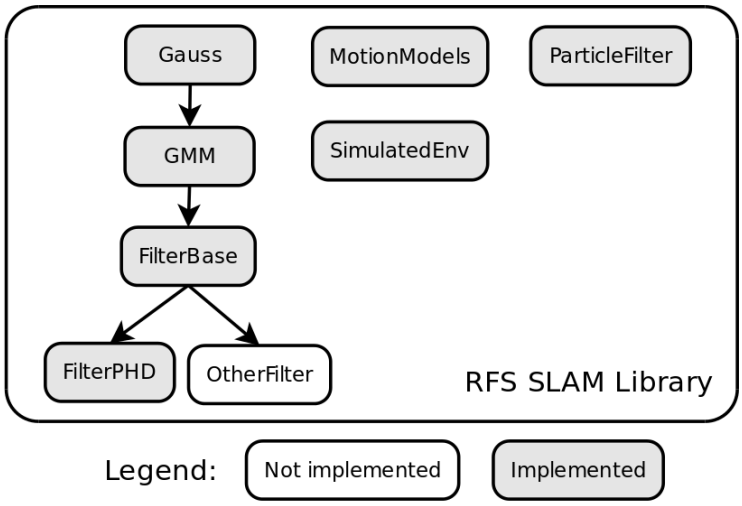



Figure 9: RFS library class hierarchy

### 5.3.1 GMM Library

Gaussian Mixture Model library is core component of our implementation. It is used for representing Guassian Mixture Model which is necessary for the representation of the intensity function as was described in 4. Because there were no 3rd-party library meeting our requirements the *GMM* library was implemented. It consist of two adaptable classes: one for Gaussian representation and second for Gaussian Mixture Model.

**Gaussian Representation** *Gauss* is a class, which represents a simple N-dimensional weighted Gaussian. It is a template class which takes two classes in the input: class representing a mean value and class representing a covariance matrix. If clases of Eigen library [4] for mean (*VectorXd*) and covariance (*MatrixXd*) are used the number of dimensions is arbitrary. Note, that using fixed size matrices (*Vector2d*, *Vector3d*), as in our experiments, is better from the performance point of view. Methods for accessing/settings the Gaussian properties are implemented as well as methods for computing eigen values, probability density function in some state or Mahalanobis distance computation w.r.t. another *Gauss* instance.

**Gaussian Mixture Model** Internally, Gaussian Mixture Model class consists of a vector of Gaussians described in the previous section. Thus it requires two template classes as an input. To increase performance, the optional argument for size of preallocated memory can be adjusted in class constructor. Methods for adding or removing Gauss from the model are implemented as well as the computation of the probability density at a specified position. Moreover, the Gauss merging and pruning required for Gaussians management in the RFS mapping is implemented.

### 5.3.2 Filter Library

Several Random Finite Set Filters exist and, even only PHD filter was implemented, the extensions for another filters is expected in the near feature. With respect to that, the implementation was divided into the abstract class, where majority of helping functions are stored and the filter implementation. All RFS Filters are supposed to derive from base class and overwrite pure virtual functions for prediction and update.

**Filter Base** Abstract class *FilterBase* consists of many helping functions and most of them can be overwritten by the special pointer or in the filter implementation itself. The example of function overwriting by the pointer is the field of view method, which can be replaced by any user specified function. If it is not overwritten, the FoV can be specified by the four parameters: maximum visible distance, minimum visible distance, horizontal angle and vertical angle. Note that these four parameters are enough for most of the sensors but possibility of function overwriting make the implementation more adaptable for non standard cases. Probability of detection and clutter rate computation can be overwritten by the function pointers as well.

Moreover, the class contains pure virtual functions for prediction and update which has to be implemented for each filter independently. Multi sensor functions for prediction and update are only necessary functions which have to be overwritten. Single sensor scenario wrapper is implemented in the *FilterBase* by default. The assumption is that single sensor scenario has same implementation as multi sensor but with number of sensors equal to one. That assumption can be obeyed if the virtual functions for single sensor prediction and update is overwritten in derived class.

*FilterBase* is a template library and requires specification of the robot state class, measurement vector class and covariance matrix class. The robot state can be one of the affine transformations types: *Affine2d* or *Affine3d*. The only non intuitive restriction for dimension of the state space is that dimension of the robot state has to be smaller or equal to dimension of the measurement vector. Of course covariance matrix has to have dimension comparable to measurement vector so the observation model make sense. All mentioned restrictions are checked during the compilation of the library.

**Filter PHD** The only available implementation of the RFS Filters is the Probability Hypotheses Density Filter, which is implemented in the class named *FilterPHD*. It derives from *FilterBase* and only overwritten functions are multi sensor scenario prediction and update. The iterative solution is used.

### 5.3.3 Particle Filter Library

One of the most common filters in the robotics is the Particle Filter (PF). In our implementation the PF is a template library which requires a particle class in the input. The only restriction for the particle class is that it is clonable (i.e. constructor which takes an instance of that class in the input is implemented). The weighting strategy for particles weighting has to be specified using the function pointer, where function takes a particle in the input and returns a double number (weight). Wrapper for weight computation and functions for resampling and normalization are implemented in the PF library. Moreover, the function for maximum weighting particle is implemented which can be used for example for maximum a posteriori trajectory estimation. Because there is no restriction about the variables in the particle class, the expected a posteriori trajectory has to be evaluated in the post processing and is not a part of the library.

### 5.3.4 Motion Models Library

To propagate particles in the state space, *MotionModels* library was designed. It is a class which contains four parameters used to tune the odometric motion model from section 2. Moreover, function which returns the propagated position and takes the start position and robot input in the form of Affine 2D transformation is implemented.

### 5.3.5 Simulated Environment

To verify correctness and evaluate properties and performance of implemented RFS filters, simulated environment is used. The dimensions of the state space and the number of static landmarks are required in the input. Both, two and three dimensional state spaces are supported. Sensor field of view can be adjusted as well as clutter rate of the measurement model to simulate real sensor as well as possible. Appropriate functions for observation extraction based on the robot position are implemented. Moreover, static function for circular trajectory generation exists in the class and is used for virtual mapping and MVSLAM verification.

## 5.4 Nodes

Following scenarios are implemented using the proposed libraries. All scenarios are separated nodes which are runable inside the Robot Operating System.

**Virtual Mapping** Node implemented to evaluate properties of the PHD Filter implementation for the purpose of robotic mapping. Several clutter rates and observation noises are simulated and evaluated in parallel. The intensity function (represented by the GMM) is saved for each time stamp during the virtual mapping. Saved values are then post processed using *Matlab* to evaluate properties of mapping. Detailed description as well as results are shown in simulated environment section.

**Virtual MVSLAM** Various motion noise and particle weighting strategy are evaluated in this node. The position and weight of the each particle is saved for each time stamp, during MVSLAM is running. Particle processing is parallelized using *OpenMP* API.

### 5.4.1 Dataset collection

Real life scenario was processed offline and to do that a tool for dataset collection is implemented. Node *collect_dataset* is subscribed to the topics (i.e. sort of pipe in ROS) where point clouds and laser scans are published. Scans and point clouds are processed independently, both in separated thread. Data are saved in the specified folder ( */.ros/rfs_collect/{type}* by default) where type is either *pc*, for point clouds, or *scan* for laser scans. Saved data have names which consist of UTC time stamp. Point clouds are saved in the binary mode using *PCL* library [17] and scans are saved using ROS serialization tools for message transportation. Real dataset was collected using this node and then post processed to estimate trajectory. The robot inputs as well as ground truth positions were measured by another sensors and tool for collecting these type of data is not part of our library.

### 5.4.2 Real MVSLAM

The multi-vehicle SLAM node implementation can be used to estimate trajectory if dataset was collected in advance. Features are detected automatically in the point clouds during estimation and are stored in a separate directory (next to *pc* and *scan* directories). Detected features are stored only for the purpose of caching, if several values of MVSLAM parameters are tuned. The presence of the timestamped robot inputs in the form of *Affine2d* transformation is assumed. With these inputs, the Particle Filter and RFS Filter libraries are used to estimate positions and results are stored for the purpose of analysis. Note that only small modification is necessary to switch from offline MVSLAM to online implementation but because of further analysis of the results it is not necessary for us.

# 6 Simulated Experiments

To evaluate properties and performance of implemented RFS MVSLAM, a virtual environment was designed. The main advantage is that the environment is well defined and static. Though implementation is adaptable to any dimension, two dimensional state space for landmarks and the robot was used. The main reason for that is better visualization of a created map and estimated trajectory with comparison to the 3D state space. Moreover, the behavior of implementation in the real three dimensional environment is described in the next section. All experiments were simulated on the CPU Intel(R) Core(TM) i7-2600 CPU @ 3.40 GHz with the 8 cores. Number of cores has no influence on the mapping algorithm performance, but multi threading is used for the SLAM experiments. Two kind of experiments were done, one for evaluating properties of the PHD mapping and second dealing with MVSLAM.

## 6.1 PHD Mapping

Performance and precision of the described mapping algorithm is shown first. Virtual environment is described and used and several observation noises and clutter rates are tested. Presence of the ground truth trajectories is assumed for the purpose of PHD mapping.

### 6.1.1 Experiment Setup

The designed 2D environment consists of 50 static landmarks uniformly sampled in the state space. The dimensions of the state space was limited to the $20 \times 20$ m. Two robots, equipped with the same sensor, are traversing circular trajectory as is visualized in Fig. 11. Both robots are moving on discretized (100 positions per loop) trajectory with total number of positions equal to 200 per robot.

**Sensor Simulation** The same sensor, measuring noised positions of the landmarks in a limited filed of view is used for both robots. Sensors were placed to the center of the robot so the additional transformation of the measurements are not necessary. Field of view was set to 4 m for a visible distance and 270 degrees for an angle (Fig. 10). Such a parameters of FoV are common for two dimensional laser range finders. Measured observations are noised and noise is simulated by standard deviations $\sigma_{ox}, \sigma_{oy}$ (i.e. observation x and y). Additionally, the clutter rate of the simulated sensors is modeled by the parameter $\lambda_o$, which represent average number of uniformly sampled clutter measurements in the squared meter in the sensor field of view. The environment with the concatenated measurements along the ground truth trajectory is shown in Fig. 11. The performance and precision of the implemented methods are evaluated for various noise and clutter settings.

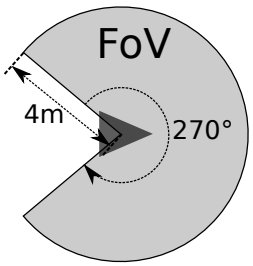
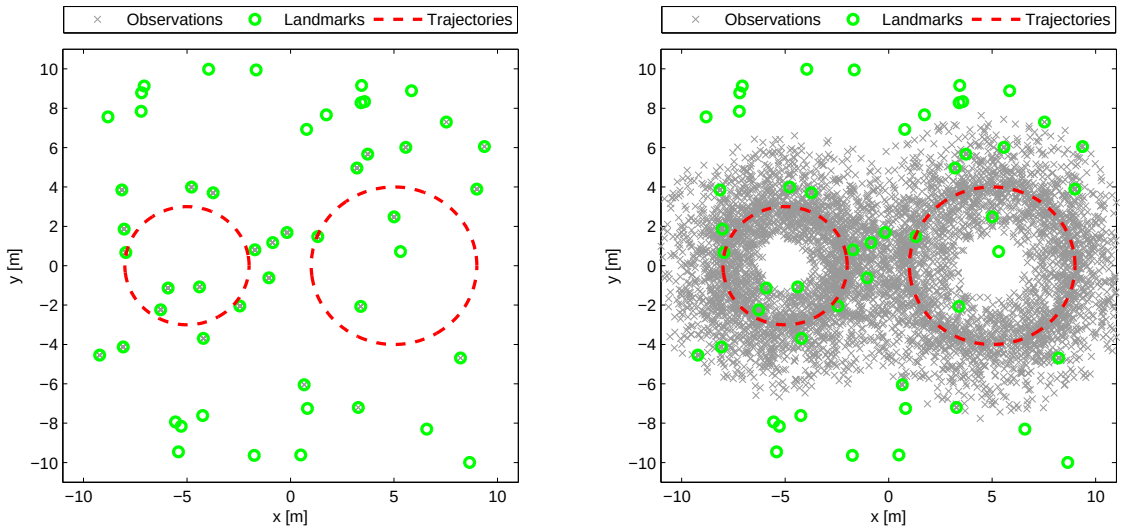



Figure 10: Simulated robot filed of view




(a) $\lambda_o = 0$ (b) $\lambda_o = 0.5$

Figure 11: Simulated environments examples

### 6.1.2 Map Precision Evaluation

To quantify precisions for different environment properties, the metric proposed in [11] is used. Metric uses Hungarian assignment algorithm to optimally assigns each feature to its ground truth while evaluating an error distance. Moreover, the wrong estimation of the number of features is considered in the proposed metric. For feature based map the estimation error is given by:

$$d_M(c)(\mathcal{M}_1, \mathcal{M}_2) = \sqrt{\frac{1}{|\mathcal{M}_1|}\left(\min_{j \in perm(\mathcal{M}_1)} \sum_{i=1}^{|\mathcal{M}_2|} d(c)(m_1^i, m_2^{j(i)})^2 + c^2 \cdot (|\mathcal{M}_1| - |\mathcal{M}_2|)\right)},$$

where $\mathcal{M}_i$ is the feature based map, $|\cdot|$ stands for number of features in map and $|\mathcal{M}_1| > |\mathcal{M}_2|$ is assumed. The function $d(c)(m_i, m_j)$ is evaluated as the minimum of the cut-off

parameter $c$ and the Euclidean distance between $m_i$ and $m_j$:

$$d(c)(m_i, m_j) = \min\left(c, \parallel m_i - m_j \parallel\right).$$

In simulated experiments one of the map is constructed using ground truth features which was at least once visible by the robot. The second map is constructed from the estimated map $D_k$, where the number of features is evaluated as the integration over the map and the mean value of the $|\mathcal{M}|$ highest weighted Gaussians is used for the feature location.

### 6.1.3 Results

The influence of the various observation noises and clutter rates is shown in Fig. 12. Estimated features as well as concatenated observations and ground truth for some of the experiments are shown in Fig. 13. In can be seen, that some of the landmarks were not detected because of the observation noise and some of the features were wrong estimated because of presence of clutter measurements. Note, that for the worst case simulated scenario there were averagely 36 observations per scan from which only 4 were correct (Tab. 1).

Processing time of experiments is measured for each experiment individually and results are shown in Table 1. It can be seen that processing time increases dramatically with the number of Gaussians in the map and the number of observations (which is related to the clutter rate). The number of experiments in the table is smaller than experiments shown in figures because only interesting results were pointed out.

Table 1: Performance analyses of the PHD mapping

| $\lambda_o$ [$m^{-2}$] | $\sigma_{ox}, \sigma_{oy}$ [m] | Average Number of Gaussians | Average Number of Measurements | Processing Time [s] |
|---|---|---|---|---|
| 0 | 0 | 24.105 | 4.26 | 0.13494 |
| 0 | 0.01 | 25.015 | 4.26 | 0.055684 |
| 0.1 | 0 | 349.36 | 12.26 | 4.1883 |
| 0.1 | 0.01 | 388.95 | 12.26 | 3.0169 |
| 0.3 | 0 | 1072.1 | 24.26 | 35.233 |
| 0.3 | 0.01 | 1153.2 | 24.26 | 39.93 |
| 0.5 | 0 | 2018.4 | 36.26 | 93.885 |
| 0.5 | 0.01 | 2060.6 | 36.26 | 80.38 |

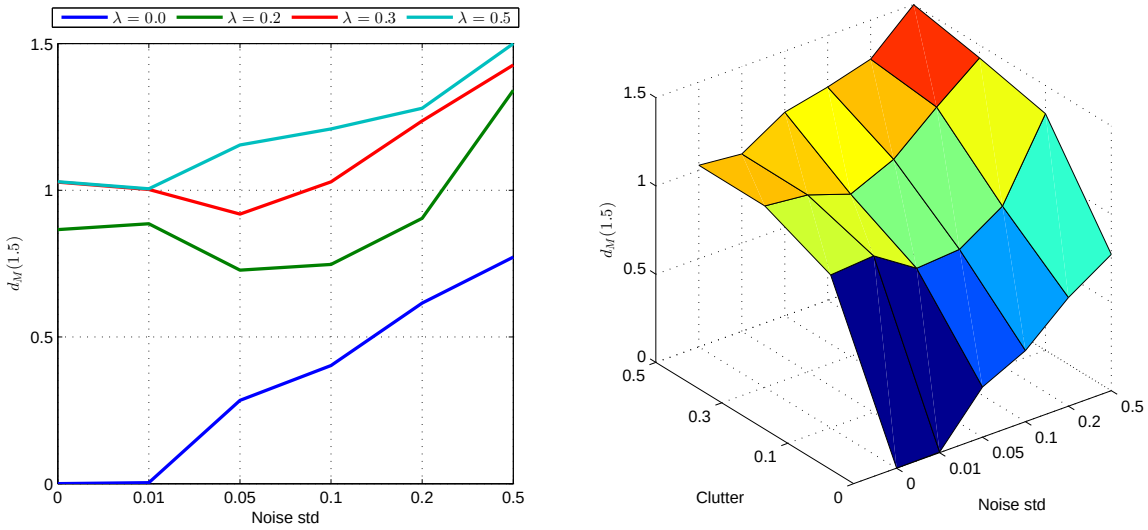
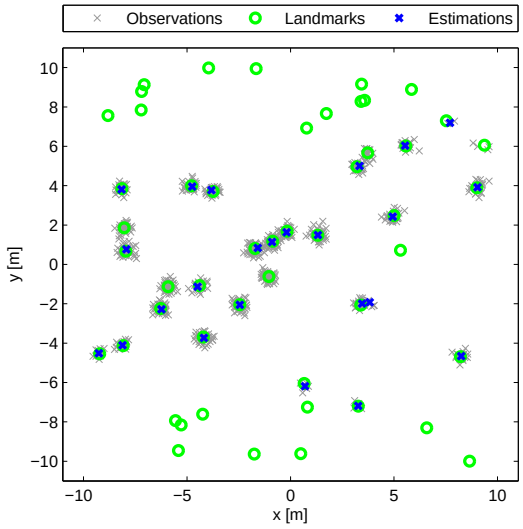
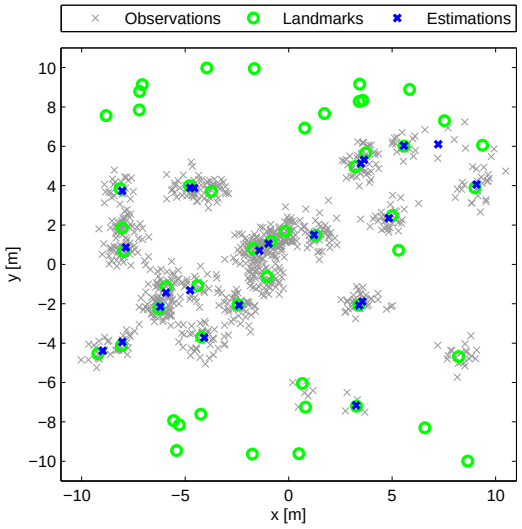


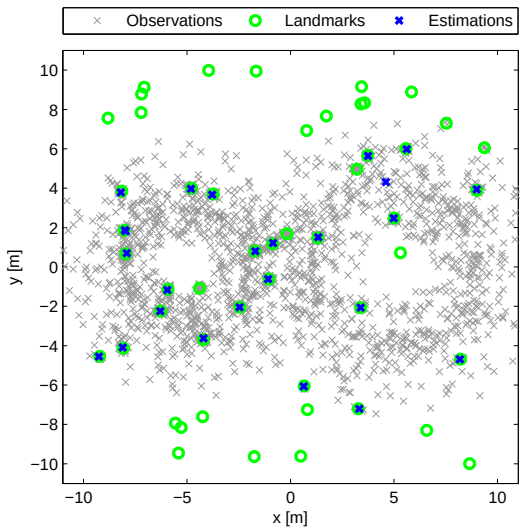

Figure 12: Virtual mapping precision evaluation



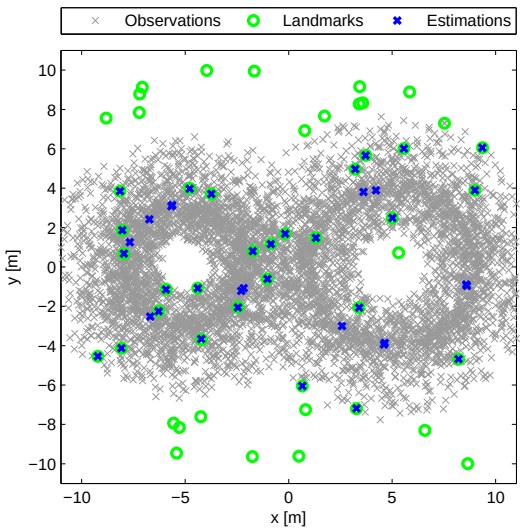

(a) $\sigma_{ox}, \sigma_{oy} = 0.2, \quad \lambda_o = 0$




(b) $\sigma_{ox}, \sigma_{oy} = 0.5, \quad \lambda_o = 0$




(c) $\sigma_{ox}, \sigma_{oy} = 0.1, \quad \lambda_o = 0.1$




(d) $\sigma_{ox}, \sigma_{oy} = 0.01, \quad \lambda_o = 0.5$

Figure 13: Virtual mapping estimated features examples

## 6.2 MVSLAM

The same circular motion as described in the section 6.1.1 is used for evaluating properties of MVSLAM with unknown initial correspondences. Both robots are used with total number of 500 particles for the position representation. Trajectory is estimated using expected a posteriori method. Several values of motion noises are tested on the static map with constant clutter rate and observation noise. Additionally, the difference between weighting strategies is shown (empty vs. single feature map strategy). MVSLAM was performed from the view of the first robot and precisions of estimated trajectories are studied independently for both robots. The estimated relative transformation was precise and noiseless, when robots met each other.

**Motion Simulation** The same simplified odometric motion model is used for both robots. Two constants, $\sigma_{ml}, \sigma_{mr}$ (i.e. motion linear and rotational), are used to model uncertainty of motion such a wheel drift. Using this constants the output from the vehicles is computed as:

$$u_l = \mathcal{N}(u_{gl}, \sigma_{ml}) \qquad u_r = \mathcal{N}(u_{gr}, \sigma_{mr}),$$

where $\mathcal{N}(\mu, \sigma)$ is Gaussian noise with mean value $\mu$ and standard deviation $\sigma$ and prefix $g$ in variable index stands for ground truth value.

### 6.2.1 Precision Evaluation

The trajectory based comparison of SLAM [1] is used to quantify precision of estimated trajectories. In that metric the relative transformations are compared instead of the global poses of the robot. Comparing the relative transformations quantify the quality of estimation better then poses comparison. The reason why trajectory based comparison was designed is shown by an example: Let consider a model situation where one of the wheel on the robot drift at the beginning of the motion. Drift will lead to wrong rotation estimation and because the map was not created in advance there is no way SLAM can correctly estimate the trajectory. Mapping and SLAM as well will continue from different start position with respect to ground truth. Even if the rest of the relative estimations are correct the position of the robot is different with respect to the global ground truth. Moreover, that kind of error can be incorporated into the estimation during the whole SLAM process. The source of the error can be for example the non-static largest weight feature location or not correctly estimated position of the sensor with respect to the motion model.

Used metric is based on the relative displacement between poses and computed as:

$$d_T = \frac{1}{N} \sum_{i}^{N} \left( \delta_1^i \ominus \delta_2^i \right)^2,$$

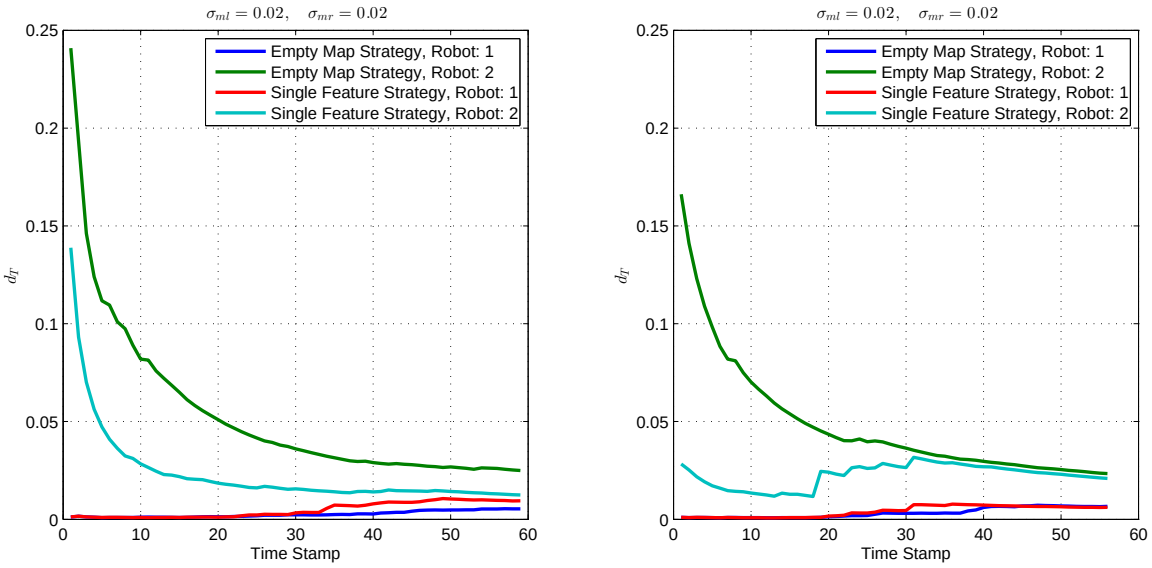
where $\delta_j^i$ is the $i$-th relative transformation, $N$ is total number of transformations and $\ominus$ stands for inverse transformation composition. The subindex $j$ stands for first or second

trajectory. One of the trajectory is ground truth and second is the estimation of the MVS-LAM in our experiments. The equation can be divided into the translational and rotational part:

$$d_T = \frac{1}{N}\sum_{i}^{N} trans\left(\delta_1^i \ominus \delta_2^i\right)^2 + rot\left(\delta_1^i \ominus \delta_2^i\right)^2. \tag{3}$$

### 6.2.2 Results

Circular trajectories were traversed by the both robots three times with total number of poses equal to 60. Robots meets each other in the pose number 27 (pose number was chosen manually and in this pose robots are close to each other so the detection is theoretically possible). Various motion noises were simulated and some of the estimated trajectories are shown in Fig. 15. Complete table of the estimated trajectories error as well as the performance analyses is shown in Table 2. The trajectory error during the time for the both strategies is shown in Fig. 14. It can be seen (14a) that the error for second robot is relatively big. The reason for that is a small number of particles in the PF (500) because when three robots (2 real + 1 virtual) are present in the scenario with a 500 particles it is less then 8 particles per robot. It can be seen (14b) that the error is reduced with higher number of particles ($25^3$).

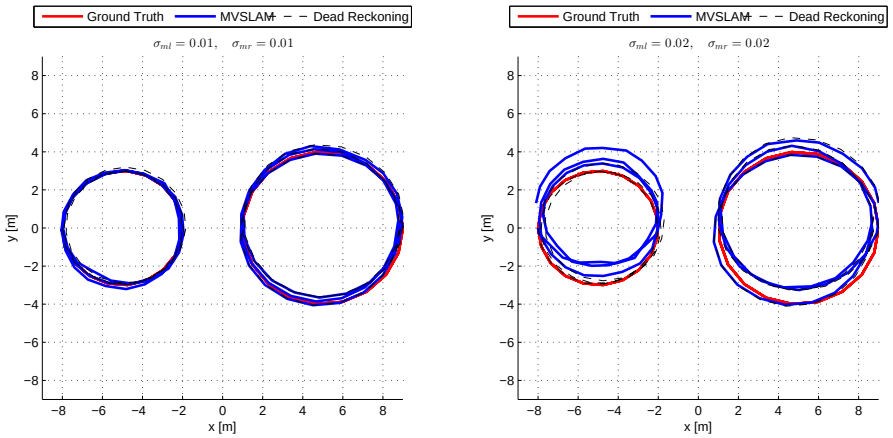



(a) Number of particles: 500 (b) Number of particles: 15625

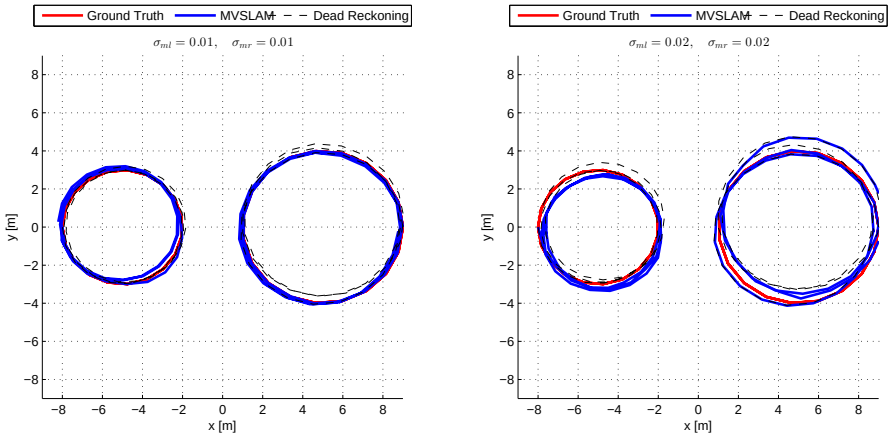
Figure 14: Trajectory error time analyses

Table 2: Performance and precision MVSLAM analyses

| $\sigma_{ml}$ | $\sigma_{mr}$ | Strategy | First Trajectory Error | Second Trajectory Error | Processing Time [s] |
|---|---|---|---|---|---|
| 0.01 | 0 | Empty Map | 0.0001 | 0.0003 | 1.0494 |
| 0.01 | 0 | Single Feature | 0.0001 | 0.0001 | 1.0443 |
| 0.01 | 0.01 | Empty Map | 0.0018 | 0.009 | 2.9985 |
| 0.01 | 0.01 | Single Feature | 0.0011 | 0.0024 | 1.3333 |
| 0.01 | 0.02 | Empty Map | 0.0127 | 0.0331 | 3.8566 |
| 0.01 | 0.02 | Single Feature | 0.0104 | 0.0139 | 2.1057 |
| 0.02 | 0 | Empty Map | 0.0005 | 0.001 | 1.3694 |
| 0.02 | 0 | Single Feature | 0.0005 | 0.0005 | 1.0448 |
| 0.02 | 0.01 | Empty Map | 0.0014 | 0.0068 | 3.4671 |
| 0.02 | 0.01 | Single Feature | 0.0014 | 0.009 | 1.2776 |
| 0.02 | 0.02 | Empty Map | 0.0134 | 0.025 | 3.9399 |
| 0.02 | 0.02 | Single Feature | 0.0095 | 0.0125 | 1.5148 |




(a) Empty map strategy




(b) Single feature strategy

Figure 15: Virtual MVSLAM examples of estimated trajectories

# 7 Experiments on Real Dataset

To shown robustness and correctness of the implemented algorithms it is necessary to perform real environment experiments. Real life experiments are typically more noised and non static at least the experiment in the outdoor. To experimentally evaluate the correctness of the MVSLAM the real outdoor dataset was used. Dataset was collected in advance so the various settings of the MVSLAM parameters can be tested and evaluated to produce correct results. Namely, the motion and the sensor parameters as well as number of particles and weighting strategies have to tuned because they were not estimated in advance.

## 7.1 Collected Dataset

Experiment used for dataset collection was measured in the Faculty of Electrical Engineering and Communication, Brno University of Technology, Department of Control and Instrumentation with a help of doc. Ing. Luděk Žalud, Ph.D. and Ing. Tomáš Jílek. The reason why the experiments were held there is presence of differential GPS sensors which was used to estimate ground truth trajectories. Because of technical limitation, only one robot was used to collect dataset and for the purpose of MVSLAM the trajectory of the robot was divided into the several parts. Whole traversed ground truth trajectory is shown in figure 16. Relative transformation is estimated from the ground truth trajectory when robots meets because only single robot was used.

### 7.1.1 Testbed Description

Robot [20] used for dataset collection was equipped by the two sensors measuring the environment. One of the sensors was laser range finder Hokuyo measuring the distances to the closes obstacles in two dimensions. The second sensor was Xtion Pro, which measures 3D distances to the closest obstacles using structured light technology. Because structured light sensors are influenceable by the sun light, the dataset was collected during night. Moreover, the positions of the robot were measured by the differential GPS sensor with the theoretical precision under a centimeter. Velocity of wheels was recorded from the operator inputs (i.e. requested speed was stored). Internal controller was responsible for velocity regulation to achieve requested velocity. Whole testbed is shown in Fig. 17.

### 7.1.2 Collected Data Description

Since GPS data, robot inputs and the sensors data were measured by different computers, the UTC time stamps are used to find the data synchronization. GPS data provides precise time stamp and all computers were synchronized with the external time server few minutes before the measurement. Because of small sensors field of view, the number of

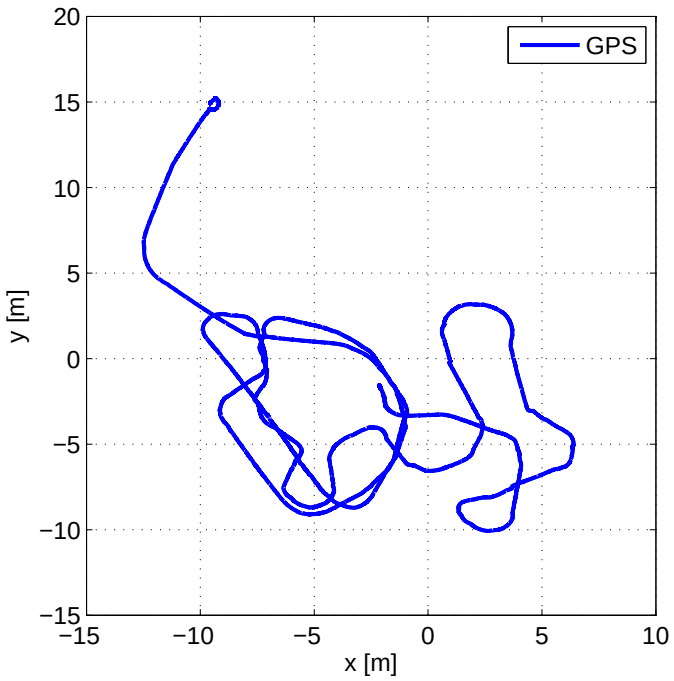



Figure 16: Ground truth obtained from GPS

landmarks in the environment was increased artifically by the people standing near the robot trajectory.

**Sensor Positions** Since sensors was not placed in the robot center, the additional post processing is required to transform measured data. Positions were not explicitly calibrated and only rough estimations exist which are shown in Fig. 18. The sensors transformations were estimated to:

$$
\begin{aligned}
T^{Base}_{Xtion} &= \{\begin{pmatrix}-0.225 & 0.175 & 0.630\end{pmatrix}, \begin{pmatrix}-\frac{\pi}{2} & 0 & -\frac{\pi}{2}\end{pmatrix}\},\\
T^{Base}_{Hokuyo} &= \{\begin{pmatrix}-0.225 & 0.175 & 0.70\end{pmatrix}, \begin{pmatrix}-\frac{\pi}{2} & 0 & -\frac{\pi}{2}\end{pmatrix}\},\\
T^{Base}_{GPS} &= \{\begin{pmatrix}0.225 & 0 & 0\end{pmatrix}, \begin{pmatrix}0 & 0 & 0\end{pmatrix}\},
\end{aligned}
$$

where each transformation consists of translation part $x, y, z$ and rotational part represented by $roll$, $pitch$, $yaw$.

**Measurement Management** Xtion Pro sensor produce measurements with a rate of 30 Hz and laser scanner with a rate of 15 Hz. To reduce memory requirement, measurements was stored with maximum frequency of 2 Hz. The name of the measurement consist of the time stamp when measurement was taken. Moreover, measurements from different sensors are stored in different folders to simplify dataset management.

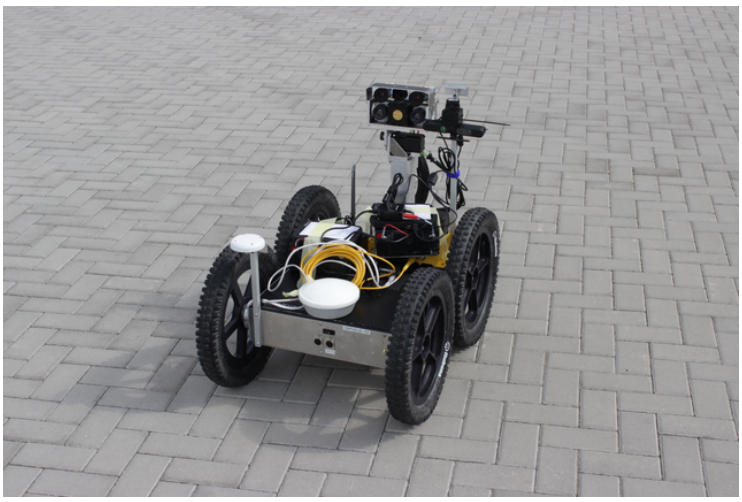


Figure 17: Testbed used for data collection

**Inputs and GPS** Robot inputs were measured in form of relative wheels velocity with respect to maximum possible velocity. All inputs are stored in one file and each row of the file consists of time stamp and relative velocity of left and right wheel. Ground truth trajectory is estimated from the GPS data and because only the positions were measured, bearing is estimated ad-hoc from the positions differences.

## 7.2 Real MVSLAM

On the small part of the collected dataset, MVSLAM algorithm is performed with the Xtion Pro sensors data used as the algorithm inputs. From the raw Xtion measurements a point cloud is reconstructed and the significant points are detected. The point cloud is a data structure used to represent collection of points.

### 7.2.1 Feature Detector

The point cloud adaptation of the SIFT keypoint detector is used to extract significant points from the measurements. SIFT keypoint detector computes Scale Invariant Feature Transform keypoints for a given point cloud which consist of points and associated intensities. Since our point cloud contains points without intensity, the height of the points are used as the intensity values. Detector is tuned to extract approximately 50 keypoints (depending on the scene) from the point cloud consisting of approximately 200000 points. Example of the point clouds with the detected features are shown in Fig. 19.

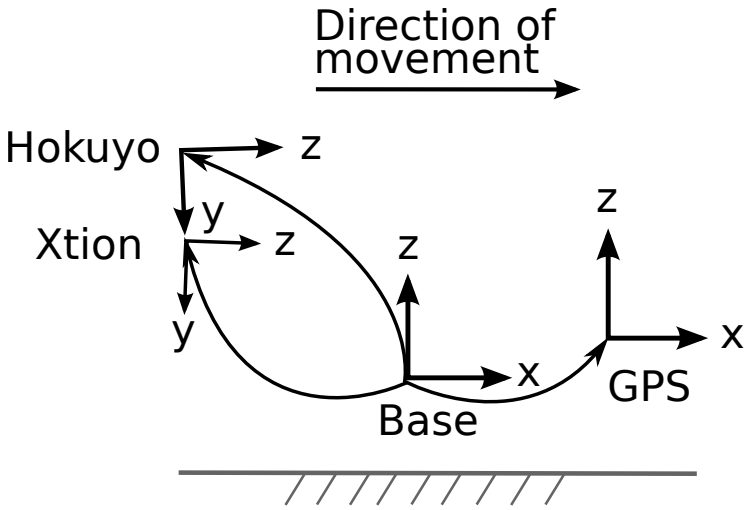



Figure 18: Sensor positions on the real robot

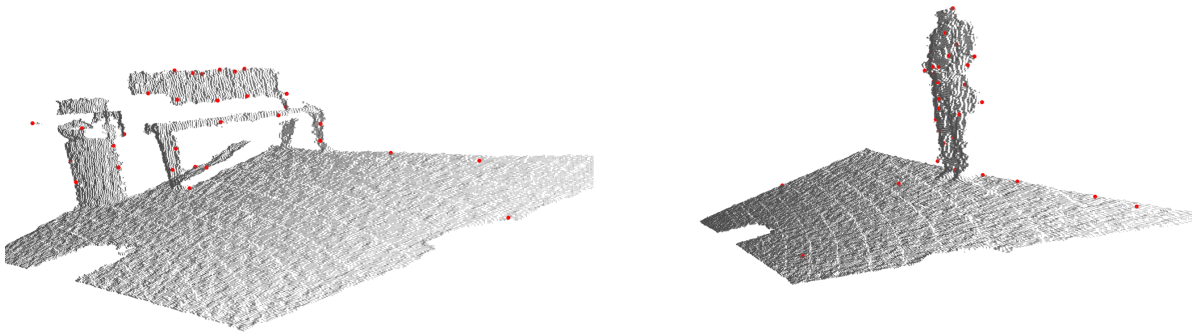


Figure 19: Features detected in the point clouds

### 7.2.2 Results

Three experiments were done. In the first two, the single robot SLAM was computed and estimated trajectories are compared to the ground truth obtained from GPS in the Fig. 21. It can be seen that the estimation of the trajectory was closer to the ground truth with respect to the dead reckoning in the first robot experiment. In the second experiment the robot commands were too noisy and thus estimation is worse. Scene, whose reconstruction was based on the estimated positions of the first robot, is shown in Fig. 20.

The distances between the ground truth and the estimated trajectories during the time are shown in Fig. 22. Distance in time $t$ is computed from estimated positions from time 1 to $t$. Note, that only relative positions were compared because raw GPS data does not contain bearing information. According to equation 3 the error computation changes to:

$$d_T = \frac{1}{N} \sum_{i}^{N} \left( trans\left(\delta_1^i\right) \ominus trans\left(\delta_2^i\right) \right)^2,$$

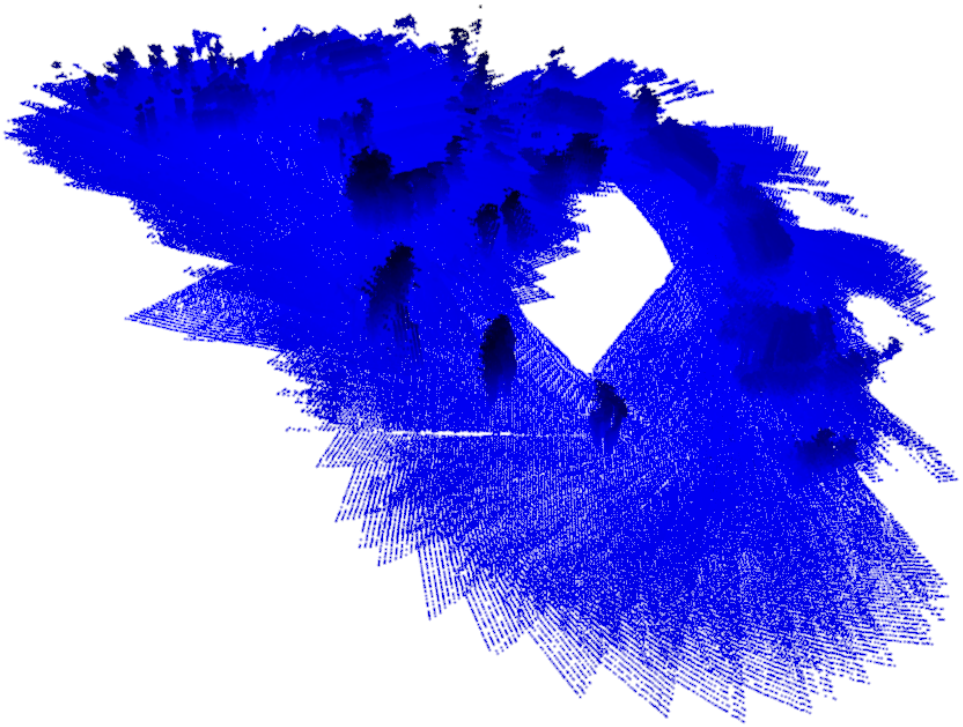
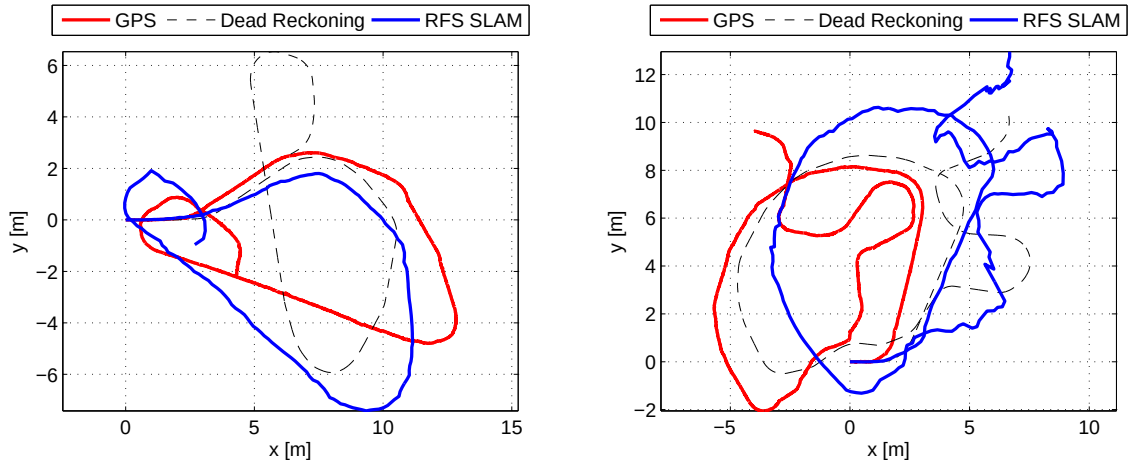


Figure 20: Reconstructed 3D scene based on the estimated trajectory

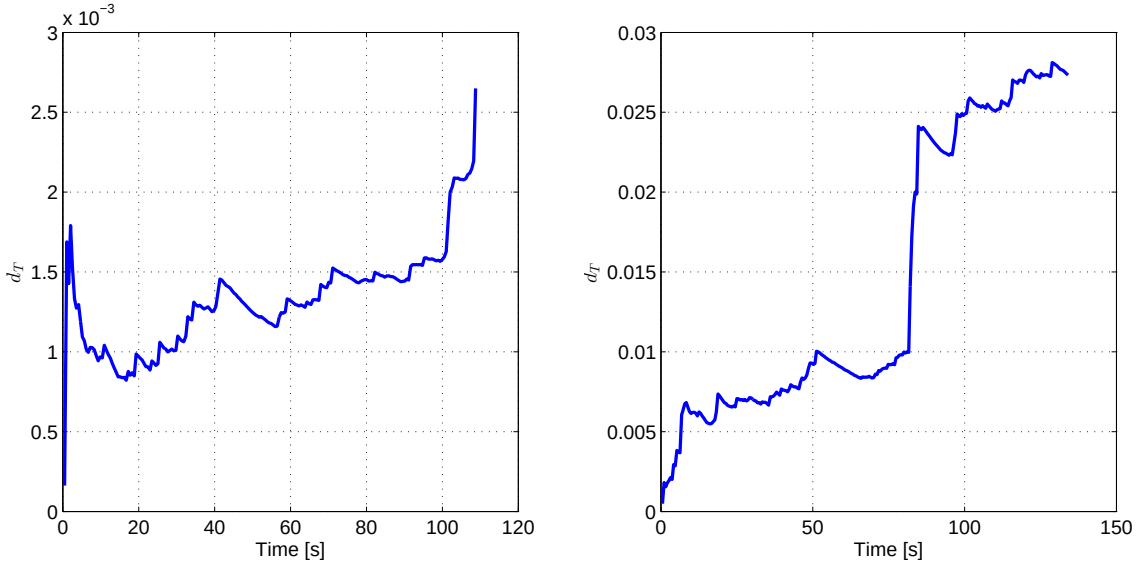
where $\ominus$ stands for euclidean distance in this case.

Second experiment evaluates the performance of MVSLAM with known initial correspondences (i.e. virtual robot is not used). Scene equal to the first two experiments was used and results are shown in Fig. 23 and 24. To conclude the results, we perform single robot as well as multi robot SLAM on the real dataset which was measured by the 3D sensor Xtion Pro. Since there were to many uncertainties in the dataset, the estimated trajectories are not perfect but some improvement with respect to dead reckoning is visible. Mentioned uncertainties include imprecise motion model, not calibrated sensor positions and possibility of unsynchronized time between computers.




(a) First robot (b) Second robot

Figure 21: Single robot SLAM estimated trajectories




(a) First robot (b) Second robot

Figure 22: Distances between estimated trajectories and the GPS

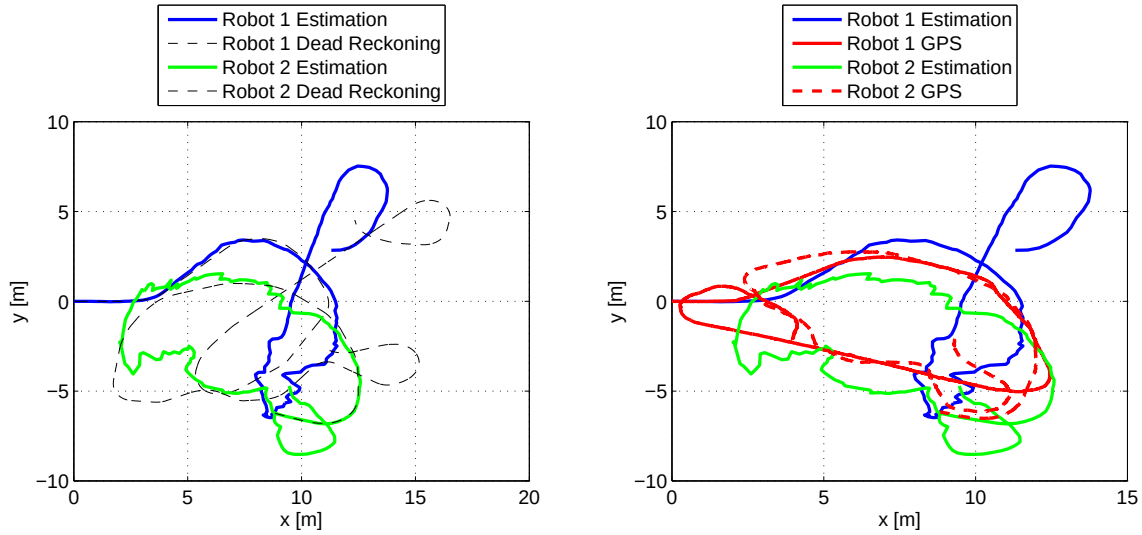



Figure 23: MVSLAM estimated trajectories

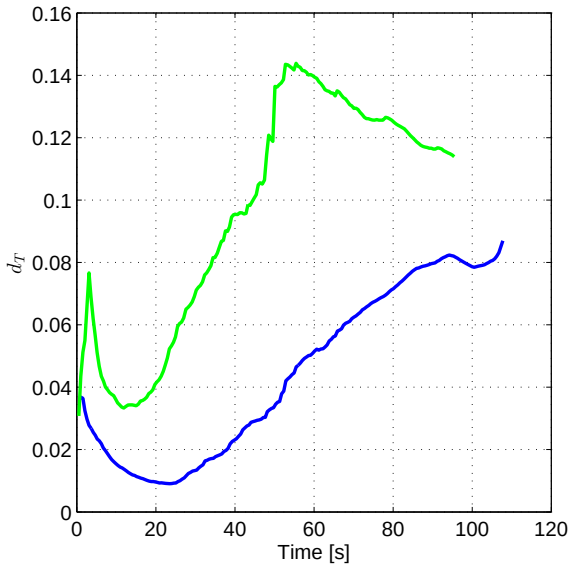



Figure 24: Distances between estimated trajectories and the GPS

# 8 Conclusion

The main goal of this thesis was to implement algorithm for Multi Vehicle Simultaneous Localization and Mapping using Random Finite Set Theory. The SLAM is one of the fundamental problem in the mobile robotics and detailed description and mathematical formulation of the SLAM was shown in the State of The Art section 2. Even many SLAM formulation exists the RFS shown promising results at least in the presence of clutter measurements. The main advantage of the RFS SLAM is possibility of measurements incorporation into the map without finding correspondences. Obeying the data association problem increases robustness of SLAM because state of the are technique for finding correspondences are not robust yet and the bad assigned correspondences have significant impact on the SLAM result.

In the section 3 the RFS theory was briefly introduced with impact on the one of the probability distribution representations called Probability Generating Functionals. Such a representation is extension of the Probability Generating Functions for multiple object density. The basic properties were studied and then used to derive one of the basic RFS filter called Probability Hypotheses Density Filter.

SLAM algorithm was designed in the section 4 using PHD Filter for measurements into the map incorporation and the Particle Filter for robot poses propagation. Mathematical concept of RFS SLAM was shown and then extended to the Multi-Vehicle SLAM scenario. So called forward-backward model for MVSLAM was used where each robot is performing single robot SLAM and virtual robot is used to incorporate past measurements from the observed robot into the map. It requires precise estimation of the relative transformation between the robots and in this thesis the presence of this transformation was assumed and not estimated automatically during the MVSLAM.

The proposed MVSLAM technique was implemented as the C++ Library and wrapped into the Robot Operating System. Detailed description of the implementation is shown in sec 5. The precision and the performances of the implemented methods was tested in the simulated environment for various environment properties in the section 6. Moreover, correctness of the implementation in the non simulated environment was shown in the section 7, where real life dataset was used.

To conclude, the main contributions of the thesis are: The library for MVSLAM was implemented and properties were tested for various simulated environment scenarios as well as for real outdoor environment. The real dataset was collected with a help of doc. Ing. Luděk Žalud, Ph.D. and Ing. Tomáš Jílek from the Brno University of Technology and is available publicly at [14]. The implemented library is available publicly as well and can be accessed via the dataset page or directly in the repository [13]. With comparison to the current state of the art technique, the three dimensional sensors were used in our experiments and multi-vehicle scenario was performed. Moreover, the technique for the MVSLAM with unknown initial correspondences was adopted and combined with the RFS mapping.

# References

[1] Wolfram Burgard, Cyrill Stachniss, Giorgio Grisetti, Bastian Steder, Rainer Kümmerle, Christian Dornhege, Michael Ruhnke, Alexander Kleiner, and Juan D. Tardós. A comparison of slam algorithms based on a graph of relations. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS'09, pages 2089–2095, Piscataway, NJ, USA, 2009. IEEE Press.

[2] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, Nov 1986.

[3] Albert Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, page 57, 1989.

[4] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[5] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[6] Michael Kass and Justin Solomon. Smoothed local histogram filters. *ACM Trans. Graph.*, 29(4):100:1–100:10, July 2010.

[7] Kurt Konolige. Improved occupancy grids for map building. *Autonomous Robots*, 4:351–367, 1997.

[8] R. Mahler. A Theoretical Foundation for the Stein-Winter Probability Hypothesis Density (PhD) Multi-Target Tracking Approach. In *Proceedings of the 2000 MSS National Symposium on Sensor and Data Fusion*, 2002.

[9] R. Mahler. The multisensor PHD filter: I. General solution via multitarget calculus. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7336 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, May 2009.

[10] J.E. Moyal. The general theory of stochastic population processes. *Acta Mathematica*, 108(1):1–31, 1962.

[11] John Mullane, Ba-Ngu Vo, Martin David Adams, and Ba-Tuong Vo. A random-finite-set approach to bayesian slam. *IEEE Transactions on Robotics*, 27(2):268–282, 2011.

[12] Ba ngu Vo and Wing kin Ma. The gaussian mixture probability hypothesis density filter. *IEEE Trans. SP*, pages 4091–4104, 2006.

[13] Vladimír Petrík. RFS MVSLAM. `https://bitbucket.org/voop/rfs_slam_catkin`, 2014.

[14] Vladimír Petrík and Kulich Miroslav. A Outdoor Dataset for the Evaluation of Depth SLAM algorithm. Dataset, Prague, Czech Republic, May 2013. http://imr.felk.cvut.cz/Research/Downloads.

[15] Michael K. Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

[16] D.A. Reynolds and R.C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *Speech and Audio Processing, IEEE Transactions on*, 3(1):72–83, Jan 1995.

[17] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *International Conference on Robotics and Automation*, Shanghai, China, 2011 2011.

[18] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. Intelligent robotics and autonomous agents. The MIT Press, August 2005.

[19] Ba-Ngu Vo, Sumeetpal Singh, and Arnaud Doucet. Sequential monte carlo methods for multi-target filtering with random finite sets, 2005.

[20] Ludek Zalud. Orpheus - universal reconnaissance teleoperated robot. In Daniele Nardi, Martin Riedmiller, Claude Sammut, and José Santos-Victor, editors, *RobuCup*, volume 3276 of *Lecture Notes in Computer Science*, pages 491–498. Springer, 2004.

[21] X.S. Zhou and S.I. Roumeliotis. Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1785–1792, Oct 2006.

# Contents of the enclosed CD

| | |
|---|---|
| *Vladimir_Petrik_DP.pdf* | The PDF version of this document. |
| *rfs_slam/* | ROS Package with the library source codes. |
| *documentation/* | LaTeXversion of this document. |
| *dataset/* | Downsampled dataset. |
| *photo/* | Selected photos from dataset collection experiment |